Bedienungsanleitung

C-Control Station

Best.-Nr. 12 51 13

Wichtig! Unbedingt lesen!

Bevor Sie die *C-Control Station* oder angeschlossene Geräte in Betrieb nehmen, lesen Sie bitte diese Anleitung vollständig durch! Sie erläutert Ihnen die korrekte Verwendung und weist auf mögliche Gefahren hin.

Für Schäden, die aus der Nichtbeachtung dieser Anleitung resultieren, besteht keinerlei Garantieanspruch und übernimmt Conrad Electronic keine Haftung.

Inhalt

Inhalt

Einleitung

Wir danken Ihnen für Ihre Entscheidung für die *C-Control Station*. Die *C-Control Station* ist ein modernes programmierbares Steuerungssystem. Die *C-Control Station* wurde von uns mit dem Anspruch entwickelt, die hohen Erwartungen unserer Kunden an Qualität und Funktionalität zu erfüllen.

Conrad Electronic GmbH
D-92240 Hirschau

Garantie

Jede *C-Control Station* verläßt das Werk in einwandfreiem und funktionsgeprüften Zustand!

Conrad Electronic bietet für die *C-Control Station* eine **Gewährleistungsdauer von 12 Monaten** . Innerhalb dieser Zeit werden eventuelle Transportschäden bei der Auslieferung, Fertigungsmängel oder Ausfälle am Gerät kostenfrei behoben.

Sollten die Leistungsmerkmale der *C-Control Station* Ihren individuellen Ansprüchen nicht genügen, nutzen Sie bitte unsere **Geld-Zurück-Garantie von 14 Tagen** . Senden Sie das Gerät innerhalb dieser Zeit ohne Gebrauchsspuren und in der Originalverpackung zur Erstattung des Warenwertes oder zur Verrechnung zurück.

Alle Fristen gelten ab Datum der Rechnung beziehungsweise des Kassenbons.

Im Modulgehäuse befinden sich keine Teile mit Servicebedarf durch den Anwender. Das Modulgehäuse **darf nicht geöffnet werden!** Im Falle einer Beschädigung des Gehäuseverschlußsiegels erlischt jeder Gewährleistungsanspruch!

Conrad Electronic übernimmt keine Haftung für Folgeschäden an Sachwerten oder Personen, die durch Anwendung der *C-Control Station* entstehen!

Service

Zu Ihrer Beratung stellt Conrad Electronic Ihnen ein kompetentes Team von Servicemitarbeitern zur Seite. Jede Anfrage wird schnellstmöglich bearbeitet. Spezialfragen werden an die Entwicklungsingenieure des C-Control Systems weitergeleitet.

Um unnötige Verzögerungen zu vermeiden, möchten wir Sie jedoch bitten, vor einer Anfrage noch einmal diese Anleitung, die Online-Hilfen der Programmiersoftware, die Text- und Beispieldateien und nach Möglichkeit die Informationsseiten im Internet zu studieren. Meist findet sich so schon die Lösung eines Problems!

Ihre Anfragen richten Sie bitte an unsere Abteilung Technische Kunden-betreuung:

Brief	Conrad Electronic GmbH TKB Computer und Meßtechnik Klaus-Conrad-Straße 1 92240 Hirschau
Fax	0180 / 53 12 119
Telefon	0180 / 53 12 116
Internet	http://www.conrad.de

Produktbeschreibung

Bestimmungsgemäße Verwendung

Die *C-Control Station* dient der programmierbaren Ansteuerung elektrischer und elektronischer Geräte. Eine andere als die bestimmungsgemäße Verwendung ist nicht zulässig.

Sicherheitshinweise

Lesen Sie diesen Abschnitt besonders aufmerksam durch! Bei Nichtbeachtung der Sicherheitshinweise besteht Lebensgefahr durch einen Stromschlag oder Elektrobrand!

1. Die *C-Control Station* ist in einem Gehäuse für DIN-Schienen-Montage („Hutschiene“) aufgebaut. Zur Gewährleistung des Schutzes vor Berührung gefährlicher Spannungen darf das Modul bei 230V~-Versorgungsspannung oder Relaisspannungen größer 24V nur in einem geschlossenen Schaltschrank oder in einem Schaltkasten mit Verblendung der Anschlußklemmen betrieben werden.
2. Über die insgesamt 35 Anschlußklemmen wird die *C-Control Station* mit anderen Geräten verbunden. Dabei ist grundsätzlich zwischen den Niederspannungsklemmen 1 bis 9 und 18 bis 35 und den Netzklemmen 10 bis 17 zu unterscheiden. Bei versehentlichem Vertauschen der Anschlüsse besteht Brandgefahr durch Kurzschlüsse, und können das Modul und angeschlossene Geräte schwer beschädigt werden!
3. Die *C-Control Station* darf nicht in Verbindung mit Geräten benutzt werden, die direkt oder indirekt medizinischen, gesundheits- oder lebenssichernden Zwecken dienen oder durch deren Betrieb Gefahren für Menschen, Tiere oder Sachwerte entstehen können. Die *C-Control Station* darf nicht in explosionsgefährdeter oder chemisch aggressiver Umgebung betrieben werden.
4. Zur Programmierung des Gerätes ist ausschließlich die mitgelieferte Software zu verwenden. Programmieren Sie das Modul nur mit Anwenderprogrammen, von deren Funktion Sie sich mit Hilfe des in der Software enthaltenen Simulators überzeugt haben.

Leistungsmerkmale

Die *C-Control Station* beinhaltet bereits alle nötigen Baugruppen, um zahlreiche Applikationen ohne Mehraufwand für Zusatzgeräte zu realisieren:

- Netzteil mit Spannungsstabilisierung
- Klemmen zum Anschluß eines Pufferakkus
- programmierbare Steuereinheit mit Mikrocontroller
- 6 programmierbare Digitalports
- 4 programmierbare Taster
- 4 programmierbare LEDs
- 2 Relais zum direkten Schalten von Geräten, die mit 230V-Netzspannung betrieben werden (Schließer, max. 6A Dauerstrom)
- 2 Temperatursensorschaltungen (linear -25°C...102,5°C, Auflösung 0,5°C) inklusive Sensoren und Verbindungsleitungen, die nahezu beliebig verlängert werden können
- 4 programmierbare Analogports zur Spannungsmessung (0...2,55V)
- serielles Interface zum Anschluß eines PCs
- Anschluß für eine DCF77-Funkuhrantenne
- Frequenzmeßport, zum Beispiel zum Anschluß eines Windrades bei einer Rollosteuerung
- eingebauter Piezolautsprecher

Mit dieser Ausstattung sind Sie in der Lage, in kurzer Zeit anspruchsvolle Steuerungs- und Regelaufgaben zu lösen.

Handhabung

Dieses Kapitel gibt einen Überblick über die Handhabung des Moduls. Die nötigen Detailinformationen entnehmen Sie bitte den nachfolgenden Kapiteln dieses Handbuches.

Die Arbeit mit der *C-Control Station* gliedert sich in drei Stufen

1. Programmieren des Moduls
2. Montage
3. Ausführen des Anwenderprogramms

Programmieren des Moduls

Die Programmierung des unmontierten Moduls erfolgt am PC-Arbeitsplatz. Alternativ kann ein bereits montiertes Modul im Schaltschrank umprogrammiert werden, wenn die nötige Verbindung zum PC hergestellt werden kann. Anderenfalls muß das Modul ausgebaut werden.

- Installieren Sie zunächst die Programmiersoftware von der der *Station* beiliegenden CD. Beachten Sie die Installationshinweise auf der CD (Datei install.txt bzw. readme.txt).
- Verbinden Sie die *Station* mit dem PC und der Spannungsversorgung, wie im Kapitel „Anschlußklemmen“ beschrieben. Schalten Sie die Spannungsversorgung ein.
- Schreiben Sie ein Anwenderprogramm, um festzulegen, was die *Station* im Betrieb tun soll. Lesen Sie dazu die Kapitel zur Programmierung des Moduls.
- Compilieren Sie das Anwenderprogramm mit Hilfe der Programmiersoftware.
- Testen Sie die Funktion des Anwenderprogramms mit Hilfe des Simulators in der Programmiersoftware.
- Übertragen Sie das getestete Programm in *Station* mit Hilfe der Programmiersoftware.

Die *C-Control/Station* ist jetzt programmiert und kann montiert werden.

Montage

Für den Betrieb muß die *C-Control/Station* auf einer DIN-Schiene montiert werden, zum Beispiel in einem Verteilerkasten, wie er zur Aufnahme von Sicherungen, Schutzschaltern und Relais in der Hausinstallation üblich ist.

- Montieren Sie zunächst das Trägersystem, zum Beispiel den Verteilerkasten.
- Ordnen Sie die *C-Control/Station* und andere Baugruppen Ihrer Gesamtapplikation im Verteilerkasten an. Die Montage erfolgt einfach durch Aufschnappen der Module auf die DIN-Schiene.
- Nun können Sie die Baugruppen verdrahten. Lesen Sie hierzu das Kapitel „Anschlußklemmen“.
- Gegebenenfalls muß Ihre Gesamtinstallation von einer zuständigen Stelle (Elektromeister) überprüft werden!
- Verblenden Sie abschließend zumindest die Netzanschlußklemmen.

Ausführen des Anwenderprogramms

Ist die *C-Control Station* programmiert und montiert, kann sie in Betrieb genommen werden, um das Anwenderprogramm auszuführen.

- Schalten Sie die Spannungsversorgung ein.
- Drücken Sie den Start-Taster.
- Nun können Sie oder andere Anwender die *C-Control/Station* in der von Ihrem Anwenderprogramm festgelegten Weise bedienen. Die *Station* arbeitet so lange nach Programm, bis die Betriebsspannung ausfällt oder der Reset-Taster gedrückt wird. Ein Neustart erfolgt wieder durch Drücken des Start-Tasters.

Bedienelemente

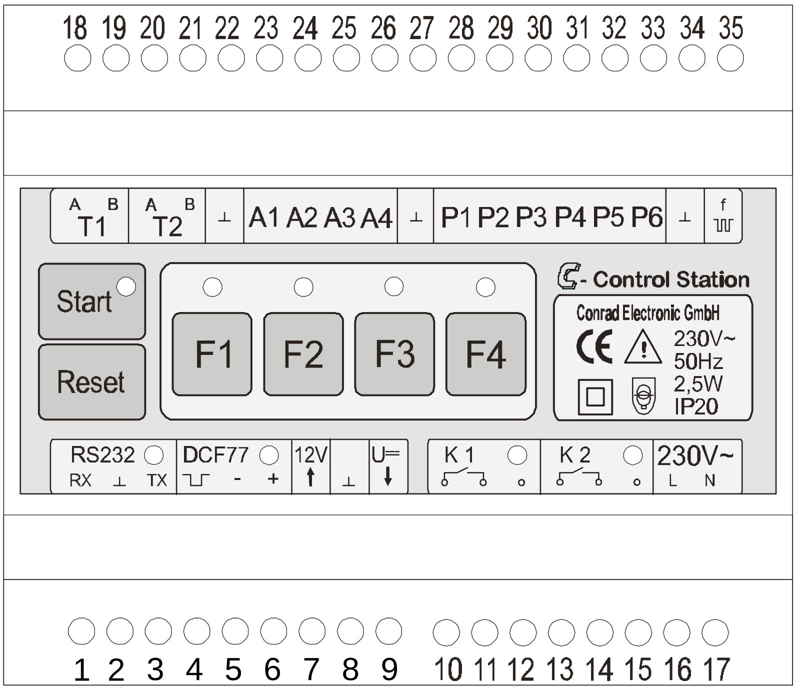


Abbildung 1

Tasten

Das Frontpanel der *C-Control Station* ist als Folientastatur ausgeführt. In diese sind insgesamt sechs Taster integriert. Drücken Sie die beschrifteten Tastfelder, bis Sie einen leichten Klick spüren.

Reset

Der Reset-Taster dient zum Rücksetzen (=„Reset“) des Moduls und wird in der Regel zum Anhalten des Anwenderprogramms oder vor der Programmierung gedrückt. Nach dem Reset befindet sich das Modul in einem initialisierten Zustand, vergleichbar dem Zustand nach Zuschalten der Betriebsspannung. Alle Anwendervariablen sind gelöscht (=0), die

interne Uhr beginnt bei 00:00:00 und wird bei Anschluß einer DCF77-Aktivantenne neu synchronisiert.

Start

Der Start-Taster startet ein Anwenderprogramm (siehe Programmierung), das zuvor über die serielle Schnittstelle in den Speicher des Moduls geladen wurde. Das Programm läuft dann solange, bis ein END-Befehl ausgeführt wird.

F1, F2, F3, F4

Die Funktionstasten können im Anwenderprogramm abgefragt und so zum Beispiel zur Auswahl von Betriebsarten, als Ein-/Ausschalter oder zur Parametermanipulation verwendet werden (siehe Programmierung).

Leuchtdioden

Die *C-Control Station* hat neun Leuchtdioden (LEDs), die durch kleine Rundfenster in der Folientastatur scheinen. Die LEDs dienen der Information des Anwenders über verschiedene Betriebszustände des Moduls. Die unterschiedlichen Farben kennzeichnen die einzelnen Funktionsgruppen: rot = System, grün = Relais, gelb = Anwender-LEDs.

LED im Start-Taster (rot)

Diese LED leuchtet, wenn gerade das Anwenderprogramm ausgeführt wird (Run-Modus), also nach Drücken des Start-Tasters bis zum END-Befehl im Anwenderprogramm, bis zum Reset oder Abschalten der Betriebsspannung.

LED für die serielle Schnittstelle (rot)

Diese LED signalisiert eine Datenübertragung auf der TX-Leitung der seriellen Schnittstelle des Moduls. Die LED ist schaltungstechnisch an die Datenleitung gekoppelt und leuchtet mit jedem übertragenen Low-Bit kurz auf, so daß ein Blinken oder Flackern der LED das Senden von Daten anzeigt.

Da beim Übertragen eines Anwenderprogramms (=„Download“) vom PC in das Modul jedes Programmbyte als Echo zum PC zurückgeschickt wird, kann über die TX-LED auch der Downloadvorgang verfolgt werden.

DCF77-LED (rot)

Diese LED zeigt bei angeschlossener DCF77-Aktivantenne den Status von Datenempfang und Synchronisation der internen Echtzeituhr des Moduls an:

die LED blink regelmäßig im Sekundentakt	Zeitdaten werden empfangen, die Synchronisation dauert je nach Empfangsqualität einige Minuten (mindestens 2...3 Minuten)
die LED leuchtet ständig	die Synchronisation ist erfolgt
die LED blink unregelmäßig	die Antenne ist schlecht ausgerichtet oder der Empfang gestört
die LED ist aus	die Antenne ist nicht oder falsch angeschlossen, falsch ausgerichtet oder der Empfang sehr stark gestört

LEDs K1 und K2 (grün)

Diese LEDs geben Auskunft über den aktuellen Schaltzustand der beiden Relais K1 und K2. Leuchtet die entsprechende LED, so ist das zugehörige Relais angezogen, der Arbeitsstromkreis ist geschlossen.

LEDs über den Funktionstasten (gelb)

Die LEDs über den Funktionstasten F1...F4 können im Anwenderprogramm ein- und ausgeschaltet werden und so zur Anzeige von Programmzuständen dienen.

Anschlußklemmen

Niederspannungsklemmen

Serielle Schnittstelle RS232

1	Datenempfangsleitung (RX), Schnittstellenkabel, weiße Ader
2	Signal Ground (Masse, GND) , Schnittstellenkabel, braune Ader
3	Datensendeleitung (TX) , Schnittstellenkabel, grüne Ader

An den Klemmen 1 bis 3 wird bei der Programmierung des Moduls das Schnittstellenkabel zum PC angeschlossen. Außerdem kann das Modul zur seriellen Datenübertragung auch während des Betriebes mit einem PC verbunden sein.

DCF77 Aktivantenne

4	Signaleingangsleitung, hier wird die low-aktive Signalleitung der DCF77-Aktivantenne angeschlossen
5	Signal Ground (Masse, GND)
6	stabilisierte Versorgungsspannung (+5V) für die DCF77 Aktivantenne

An den Klemmen 4 bis 6 kann optional eine DCF77-Aktivantenne zur Zeitsynchronisation des Moduls angeschlossen werden. Alternativ steht an Klemme 6 die stabilisierte 5V-Systemspannung zur Versorgung kleiner externer Schaltungen zur Verfügung. Beachten Sie dabei den maximal entnehmbaren Strom (siehe Technische Daten)!

Niederspannungsversorgung und Akkupufferung

7	Eingang für ein stabilisiertes 12V-Netzgerät oder einen 12V-Akku (+)
8	Ground (Masse, GND)
9	Ausgang der unstabilisierten Systemspannung, 12...24V

An Klemme 7 kann ein stabilisiertes 12V-Netzgerät oder ein 12V-Akku angeschlossen werden, um das Modul vor der Montage im Schaltschrank und ohne Anklemmen der 230V~-Versorgungsspannung (an 16 und 17) zu programmieren.

Während des Normalbetriebes im Schaltschrank mit 230V~-Versorgungsspannung kann an Klemme 7 ein 12V Blei-Gel-Akku (empfohlene Kapazität >= 1,2 Ah) zur Pufferung des Systems gegen Netzspannungsausfälle angeschlossen werden. Um die volle Schaltfähigkeit der Relais zu gewährleisten, muß der Akku gut geladen sein. Bei vorhander Netzspannung fließt ein geringer Ladestrom zum Ausgleich der Selbstentladung in den Akku. Die *C-Control Station* darf jedoch nicht dazu benutzt werden, um leere Akkus zu laden!

An Klemme 9 wird die unstabilisierte Gleichspannung des internen Netzteils herausgeführt. Sie kann zur Versorgung kleiner externer Schaltungen benutzt werden. Beachten Sie dabei den maximal entnehmbaren Strom (siehe Technische Daten)! Beachten Sie außerdem, daß der tatsächliche Spannungswert in Lastabhängigkeit deutlich über der Nennspannung von 12V liegen und bis zu 24V betragen kann!

Ports für Temperatursensoren

18	T1 A	Sensor 1, schwarzes Kabel, weiße Ader
19	T1 B	Sensor 1, schwarzes Kabel, braune Ader
20	T2 A	Sensor 2, graues Kabel, weiße Ader
21	T2 B	Sensor 2, graues Kabel, braune Ader

Die *C-Control Station* enthält elektronische Baugruppen zur Ankopplung von zwei Temperatursensoren an die A/D-Wandler des Mikrocontrollers. Als Sensoren befinden sich zwei konfektionierte Halbleiterfühler vom Typ AD592 im Lieferumfang des Systems. Diese werden über Verbindungsleitungen an die Ports T1 und T2 am Modul angeschlossen. Die interne Elektronik für jeden Sensorkanal ist auf jeweils einen der beiden Fühler abgeglichen, um ein Optimum an Meßgenauigkeit zu erreichen. Achten Sie beim Anschluß der Sensoren auf die entsprechende Farbkennzeichnung, welcher Sensor mit welchem Port zu verbinden ist (siehe Tabelle). Achten Sie außerdem auf die richtige Polung.

Der Temperaturfühler AD592 arbeitet als temperaturabhängige Stromquelle und treibt einen Strom von 1µA/K über seine Anschlüsse. Die mitgelieferten Anschlußleitungen können bei Bedarf um einige Meter verlängert werden. Um Störeinstahlungen zu vermeiden, sollte dann unbedingt ein Kabel mit Schirmleitung verwendet werden. Der Schirm der Verlängerungsleitung ist mit dem Schirm der Originalleitung zu verbinden und an der *C-Control Station* auf eine der GND-Klemmen zu legen. Achten Sie beim Anschluß der Verlängerungsleitung auf die korrekte Polarität.

GND-Klemmen (Masse)

22	GND
27	GND
34	GND

An den GND-Klemmen steht die Systemmasse als Bezugspotential für digitale und analoge Ports zur Verfügung.

Analogports

23	A1
24	A2
25	A3
26	A4

Die Analogports dienen zur Messung von analogen Spannungswerten, bezogen auf das Massepotential der *C-Control Station*. Die Meßwerterfassung erfolgt durch die A/D-Wandler des Mikrocontrollers mit 8 Bit Auflösung und liefert Werte von 0 bis 255. Die obere Referenzspannung beträgt 2,55V und entspricht dem Wandlungswert 255.

Digitalports

28	P1
29	P2
30	P3
31	P4
32	P5
33	P6

Jeder der Digitalports des Moduls kann frei als Eingang zum Erfassen eines Schaltzustandes oder als Ausgang zum Auslösen eines Schaltvorganges programmiert werden. Die logischen Pegel betragen 0...0,7V für AUS und 4,3...5V für EIN.

Frequenzmeßeingang

35	f

An Klemme 35 kann die Frequenz f digitaler Pulse gemessen werden. Das Eingangssignal ist dabei entweder ein 0/5V-Digitalsignal oder eine

zyklische Verbindung nach GND (Klemme 34), zum Beispiel durch ein Windrad, das mit jeder Umdrehung einen Kontakt schließt. Der Meßbereich mit einem Fehler <1% reicht bis ca. 30kHz.

Netzklemmen

Versorgungsspannungsanschluß

16	Phase 230V~ (L)
17	Nulleiter 230V~ (N)

Über den Versorgungsspannungsanschluß werden im Normalbetrieb alle internen elektronischen Komponenten des Moduls gespeist. Im Modul befindet sich dazu ein Netzteil mit Transformator, Gleichrichter und Spannungsstabilisierung.

Relais K1 und K2

10	K1
11	K1
12	leere Klemme
13	K2
14	K2
15	leere Klemme

Mit den Relais K1 und K2 im Modul können Geräte geschaltet werden, die mit 230V~-Netzspannung betrieben werden. Der maximal zulässige Kontaktstrom (siehe Technische Daten) darf nicht überschritten werden. Die leeren Klemmen zwischen den Relais und dem Versorgungsspannungsanschluß dienen zur Einhaltung der nötigen Sicherheitsabstände zwischen den unterschiedlichen Spannungs-potentialen.

Anschluß zum Programmieren des Moduls

Verbindung der Station mit dem PC

Um ein Anwenderprogramm vom PC in das Modul zu übertragen, sind zwei Anschlußformen bezüglich der Spannungsversorgung zulässig.

In beiden Fällen ist das Schnittstellenkabel zwischen PC und *Station* folgendermaßen anzuschließen:

- Verbinden Sie den 9-poligen Sub-D-Verbinder des der *Station* beiliegenden Anschlußkabels mit einer der seriellen Schnittstellen (COM1...COM4) des PCs
- Klemmen Sie die **weiß** markierte Ader an **Klemme 1** der Station - **RX**
- Klemmen Sie die **braun** markierte Ader an **Klemme 2** der Station - **GND**
- Klemmen Sie die **grün** markierte Ader an **Klemme 3** der Station - **TX**

Programmieren bei Niederspannungsversorgung

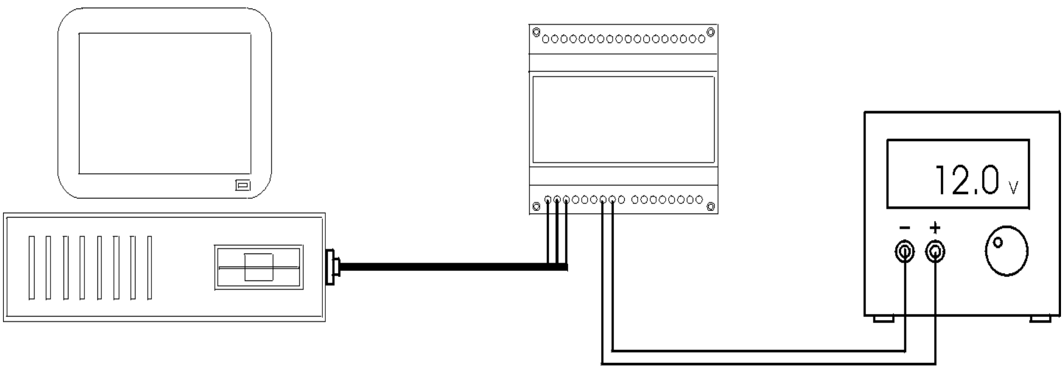


Abbildung 2

Beim Programmieren mit Niederspannungsversorgung ist ein stabilisiertes 12V-Netzgerät oder ein 12V-Akku an den Klemmen 7 (+) und 8 (-) des Moduls anzuschließen. Das Modul kann dann frei gehandhabt werden. Diese Versorgungsart eignet sich besonders zur Programmierung am PC-Arbeitsplatz, im Labor, in der Werkstatt oder im Büro.

Programmieren bei Netzversorgung

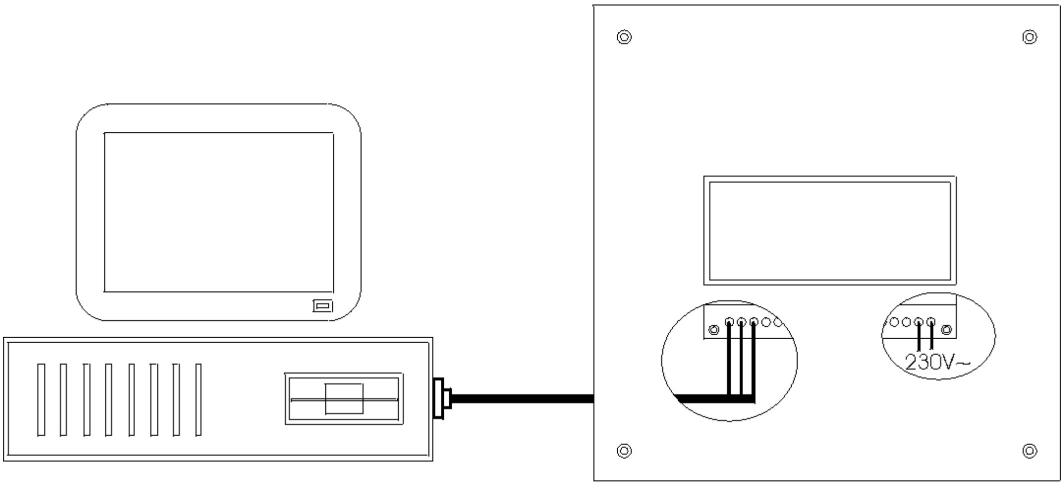


Abbildung 3

Die Programmierung mit Netzversorgung dient zur Übertragung von Anwenderprogrammen am Einsatzort des Moduls. Die *C-Control Station* ist dabei auf der DIN-Schiene montiert. Zur Gewährleistung des Berührungschutzes müssen zumindest die Netzanschlußklemmen verblendet sein.

Beim Anschluß des Programmierkabels muß die *C-Control Station* spannungsfrei sein!

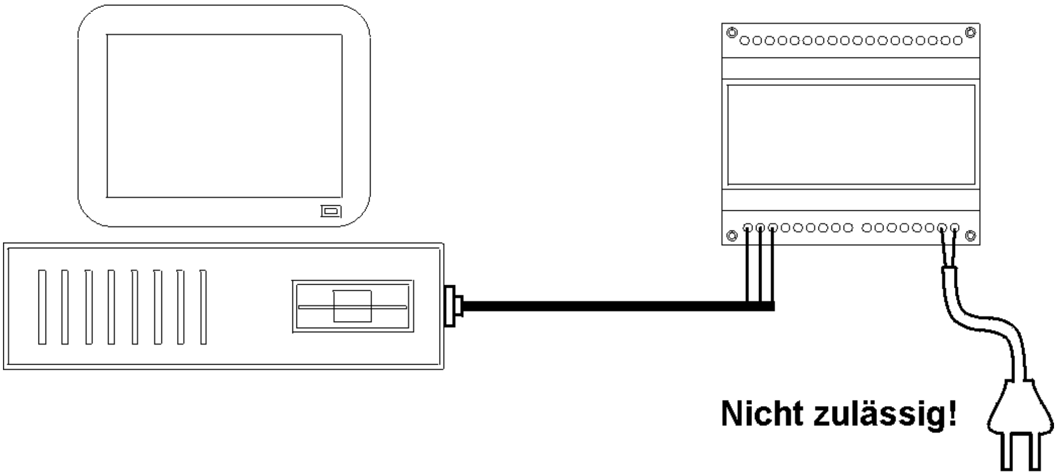


Abbildung 4

Die Versorgung eines frei am Tisch liegenden Moduls über ein Standard-Netzkabel mit angegossenem Stecker (für Geräte der Schutzklasse II) ist technisch möglich, jedoch nach den gesetzlichen Bestimmungen zum Schutz vor Berührung gefährlicher Spannungen nicht zulässig!

Systemzustände

Reset

Nach Zuschalten der Betriebsspannung oder Drücken des Reset-Tasters nimmt das Modul alle Initialisierungen vor und geht dann unmittelbar in den Standby-Modus über.

Folgende Initialisierungen werden vorgenommen:

1. Datum und Uhrzeit: 1. Januar 1997, 00:00:00
2. Vorbereitung des DCF77-Funkuhrdatenempfangs
3. Einrichten der seriellen Schnittstelle auf 9600 Baud, 8 Datenbits, 1 Stoppbit, keine Parität
4. Löschen aller Anwendervariablen (=0)

Standby-Modus

Im Standby-Modus werden permanent der Start-Taster und die serielle Schnittstelle abgefragt, um das Anwenderprogramm zu starten oder auf Steuerbefehle von einem übergeordneten Rechner (PC) zu reagieren. Im Standby-Modus kann das Modul programmiert werden (Programm-Download).

Wird der Start-Taster als gedrückt erkannt, geht das System in den Run-Modus über.

Run-Modus

Im Run-Modus werden die Befehle des Anwenderprogramms ausgeführt. Der Run-Modus endet durch Drücken der Reset-Taste, Abschalten der Betriebsspannung oder mit Ausführung des END-Befehls (bzw. des END-Blocks) im Anwenderprogramm. Nach dem END-Befehl geht das System wieder in den Standby-Modus über.

Programmierung (allgemein)

Überblick

Durch Programmierung wird die Arbeitsweise des Systems festgelegt. Dabei bietet die *C-Control Station* ein hohes Maß an Flexibilität und kann so für unterschiedlichste Zwecke eingesetzt werden. Einmal programmiert, versieht sie ihren Dienst z.B. als Temperaturregler, intelligenter Treppenlichtautomat, Alarmanlage oder Rollosteuerung. Falls einmal andere Aufgaben anstehen, kann das Modul beliebig oft umprogrammiert werden.

Im Inneren der *C-Control Station* arbeitet ein Mikrocontroller. Das ist ein kleiner Computer auf einem Chip. An diesem ist ein Speicherchip angeschlossen, der zur Aufnahme des Anwenderprogramms dient. Durch die EEPROM-Technologie bleibt das Anwenderprogramm auch dann erhalten, wenn die Betriebsspannung vom System abgetrennt wird.

Die Programmierung der *C-Control Station* erfolgt mit Hilfe einer komfortablen Software am PC, entweder in einer grafischen Entwicklungsoberfläche oder in der populären Programmiersprache BASIC.

Bits, Bytes und Words

Das C-Control System verarbeitet, berechnet und speichert ausschließlich ganze Zahlen (=„Integer“). Dabei kennt es drei verschiedene Datentypen:

Bit: Wertebereich 0 und -1 (nicht 1!); Verwendung als Zustandsspeicher (=„Flag“)

Byte: belegt 8 Bits; Wertebereich 0 bis 255; Verwendung für kleine nichtnegative ganze Zahlen, ASCII-Zeichencodes, Befehlscodes

Word (=„Integer“): belegt 16 Bits; besteht aus High-Byte und Low-Byte; Wertebereich -32768 bis 32767; universelle Verwendung

Systemressourcen

Peripherie

Letztendlich ist das Anliegen eines jeden Programms für C-Control das Schalten und Verwalten der angeschlossenen Baugruppen. Die meisten Peripherieeinheiten stehen in unmittelbarer Verbindung zu den Anschlußklemmen oder den Bedienelementen im Frontpanel.

Digitalports: Die *C-Control Station* verfügt über insgesamt 16 programmierbare Digitalports. Die Ports 1 bis 6 sind mit den Klemmen P1 bis P6 verbunden und können als Eingang oder Ausgang benutzt werden. Die Ports 7 und 8 schalten die beiden Relais K1 beziehungsweise K2. Über die Ports 9 bis 12 können die Funktionstasten F1 bis F4 abgefragt werden. Die Ports 13 bis 16 können die LEDs über den Funktionstasten ein- und ausschalten.

Analogports: Zur Erfassung analoger Meßgrößen hat die *C-Control Station* insgesamt 8 Analog-Digital-Converter-Ports mit 8 Bit Auflösung. Damit können elektrische Spannungen gemessen werden, die zum Beispiel als Ausgangssignal eines Sensors von einer bestimmten physikalischen Größe abhängen. Der Meßbereich reicht von 0 bis 2,55V in Schritten von 10 mV. An der *C-Control Station* sind die Analogports 1 bis 4 an den Klemmen A1 bis A4 herausgeführt.

Die Analogports 5 und 6 sind mit den Temperatursensoren T1 beziehungsweise T2 verbunden.

Der Analogport 7 mißt die Spannung an einem internen Punkt der Versorgungselektronik. Ist bei fehlender Netzspannung ein geladener 12V-Akku an den Klemmen 7 und 8 angeschlossen, dann beträgt der Meßwert ca. 120. Nun kann über das Sinken des Meßwertes die Entladung des Akkus während des Betriebes verfolgt werden. Sinkt die Akkuspannung unter einen kritischen Wert, können so im Anwenderprogramm rechtzeitig Maßnahmen gegen den bevorstehenden Ausfall des Systems getroffen werden.

An Analogport 8 kann der Ausfall der 230V~-Netzspannung detektiert werden. Der Port ist intern über einen Spannungsteiler mit dem „Plus“-Ausgang des Brückengleichrichters des integrierten Netzteils verbunden. Der Spannungsverlauf ist mit einem Kondensator geglättet. Sinkt der Meßwert auf 0, dann ist die Netzspannung ausgefallen. Da die Betriebsspannung des Mikrocontrollers mit einem größeren Kondensator gepuffert ist, läuft das System nach Netzspannungsausfall kurze Zeit

weiter. Bei geschickter Programmierung der Ausfallerkennung können so noch Maßnahmen zum geordneten Programmabschluß getroffen werden.

Echtzeituhr: An den Klemmen 4 bis 6 kann eine DCF77-Aktivantenne angeschlossen werden. Bei Signalempfang wird die interne Uhr des Systems automatisch auf Mitteleuropäische Zeit/Sommerzeit gestellt. Die einzelnen Zeit- und Datumsinformationen - Stunde, Minute, Sekunde, Tag, Wochentag, Monat, Jahr - können im Anwenderprogramm abgefragt und direkt oder in logischer Kombination mit anderen Bedingungen als Auslöser bestimmter Operationen verwendet werden.

Frequenzmessung: Mit dem Frequenzmeßeingang können Signale bis ca. 30kHz erfaßt und im Anwenderprogramm abgefragt werden.

Serielle Schnittstelle: Die serielle Schnittstelle dient nicht nur zum Laden eines Anwenderprogramms in die *C-Control Station*, sondern kann auch im Programm selbst zur Datenübertragung benutzt werden. Dabei ist es möglich, Daten byteweise oder als Text zu übertragen. Die Übertragungsrate kann in einigen Stufen (1200...9600 Baud) angepaßt werden.

Tonausgabe: In der *C-Control Station* ist ein Piezo-Schallwandler eingebaut. Das ist ein kleiner Lautsprecher, mit dem Tonsignale erzeugt werden können. Dabei sind Tonhöhe und -dauer programmierbar. So sind kleine Melodien, Quittungs- oder Warnsignale programmierbar, um Tastendrücke zu bestätigen oder um in bestimmten Situationen die besondere Aufmerksamkeit des Benutzers auf das Gerät zu lenken.

Programmspeicher

Das von Ihnen programmierte Anwenderprogramm wird von der Programmiersoftware in ein dem C-Control System verständliches Format übersetzt (=„compiliert“) und in den Programmspeicher der *C-Control Station* übertragen. Der Programmspeicher hat einen Umfang von ca. 8000 Bytes. Ein Befehl zum Einschalten eines der Relais belegt zum Beispiel 5 Bytes. Auch komplexere Programme benötigen selten mehr als einige Hundert Bytes. So steht stets ausreichend Platz im Programmspeicher zur Verfügung.

Der Programmspeicher befindet sich im EEPROM-Speicherchip (Typ Microchip 24C65). Aufgrund der EEPROM-Technologie bleiben alle Daten im Programmspeicher auch nach Abschalten der Spannungsversorgung erhalten.

Speicher für Datenaufzeichnungen

Die restlichen freien Bytes im Programmspeicherchip stehen dem Anwenderprogramm zur Aufzeichnung von Daten, zum Beispiel Meßwerten, zur Verfügung. Aufzeichnung und Auslesen der Werte erfolgt stets sequentiell und wordweise. Ein aufgezeichneter Wert belegt also 2 Bytes.

Die Verwaltung der aufgezeichneten Werte erfolgt nach dem Dateiprinzip. Vor einem Zugriff muß die Datei geöffnet werden. Es wird unterschieden in Öffnen zum Lesen, Öffnen zum Schreiben und Öffnen zum Anhängen neuer Daten an vorhergehende. Nach einer Zugriffssequenz muß die Datei geschlossen werden, um eventuelle Änderungen zu übernehmen.

Wie das Anwenderprogramm sind die aufgezeichneten Daten sicher vor Ausfall der Betriebsspannung. Voraussetzung dafür ist, daß nach dem letzten geschriebenen Wert die Datei geschlossen wurde.

Arbeitsspeicher

Im Verlauf eines Programms ist es oft erforderlich, bestimmte Rechenergebnisse, Zustände oder Parameter zwischenzuspeichern. Dafür bietet die *C-Control Station* insgesamt 24 Bytes Arbeitsspeicher, der über definierbare Variablen bit-, byte- oder wordweise angesprochen werden kann.

Im Gegensatz zum Programmspeicher bleibt der Inhalt des Arbeitsspeichers nur solange erhalten, wie der Mikrocontroller des Systems mit Betriebsspannung versorgt ist. Nach einem Reset oder Spannungsausfall sind alle Werte im Arbeitsspeicher rückgesetzt (=0).

Recheneinheit

Wie im Abschnitt „Bits, Bytes und Words“ bereits beschrieben, erfolgen alle Berechnungen mit ganzen Zahlen. Alle Operationen werden mit 16 Bit Breite ausgeführt (Integer-Wertebereich). Die Recheneinheit des Systems beherrscht die vier Grundrechenarten (+, -, *, /), die Modulodivision (Ermittlung des Restes bei ganzahliger Division), bitweise logische Verknüpfungen, Vergleichsoperationen, Negation, die Bestimmung des Absolutbetrages, die gerundete Quadratwurzel- und die Signumfunktion. Außerdem ist ein initialisierbarer Pseudozufallsgenerator implementiert, der nach dem Prinzip der Primzahlenmultiplikation mit anschließender Modulodivision statistisch verteilte Zahlenfolgen erzeugen kann.

Die Recheneinheit beherrscht maximal 6 Verschachtelungsebenen bei Klammerrechnungen oder vergleichbaren Operationen.

Achtung! Durch die Begrenzung auf maximal 16 Bit Datenbreite im Integer-Wertebereich kann es bei Berechnungen zu Überläufen in den Ergebnissen kommen. Außerdem findet beim Speichern eines Wertes in einer Variable prinzipbedingt eine Begrenzung auf deren deklarierten Wertebereich statt. Eventuelle Nachkommastellen von Divisionsergebnissen werden abgeschnitten.

Beispiele:

- 32767 + 1 ergibt -32768, nicht 32768
- a = -1 ergibt 255 für a, wenn a nur eine Bytevariable ist
- 10 / 4 ergibt 2, nicht 2,5

Grafische Programmierung

Überblick

Zur grafischen Programmierung steht Ihnen eine speziell auf die *C-Control Station* angepaßte Version des Programms „C-Control/plus“ zur Verfügung. Die **grafische Programmierung** ist vor allem **für Einsteiger und kleine Anwendungen** geeignet. Die Software befindet sich auf der der *Station* beiliegenden CD. Zur Installation befolgen Sie bitte die Hinweise auf der CD. Die grafische Programmiersoftware benötigt als Betriebssystem mindestens Microsoft Windows95 oder WindowsNT ab 4.0 oder entsprechende Nachfolgeversionen! Die Software kann nicht unter dem inzwischen veralteten Windows 3.xx installiert werden.

Da die Bedienelemente der Software selbsterklärend und intuitiv zu verstehen sind, wird an dieser Stelle auf eine komplette Beschreibung der Benutzeroberfläche verzichtet. Die folgenden Abschnitte versorgen Sie lediglich mit den nötigen Grundkenntnissen. Hilfestellung zur Bedienung des Programms erhalten Sie am PC durch Drücken der Taste F1 (Gesamthilfe) oder Strg+F1 (kontextsensitive Hilfe). Außerdem können Sie durch Rechtsklicken auf einen Funktionsblock und Auswahl des daraufhin angebotenen Kontextmenüpunktes „Hilfe zum Blocktyp“ eine Beschreibung der Blockfunktion bekommen.

Abbildung 5 zeigt einen Überblick über die Benutzeroberfläche zur grafischen Programmierung der *C-Control Station*. Die wichtigsten Komponenten sind beschriftet. Zur Bedienung der Software ist es erforderlich, daß Sie mit den Grundprinzipien der Bedienung von Standard-Windows-Programmen vertraut sind. Begriffe, wie „Menü“, „Statusleiste“, „Klicken“, „Doppelklicken“ oder „Ziehen mit der Maus“, „Datei öffnen“ oder „Datei speichern“ , sollten Ihnen keine Fremdworte sein. Im Zweifelsfall lesen Sie bitte noch einmal das Windows-Handbuch oder die Seiten der Windows-Online-Hilfe.

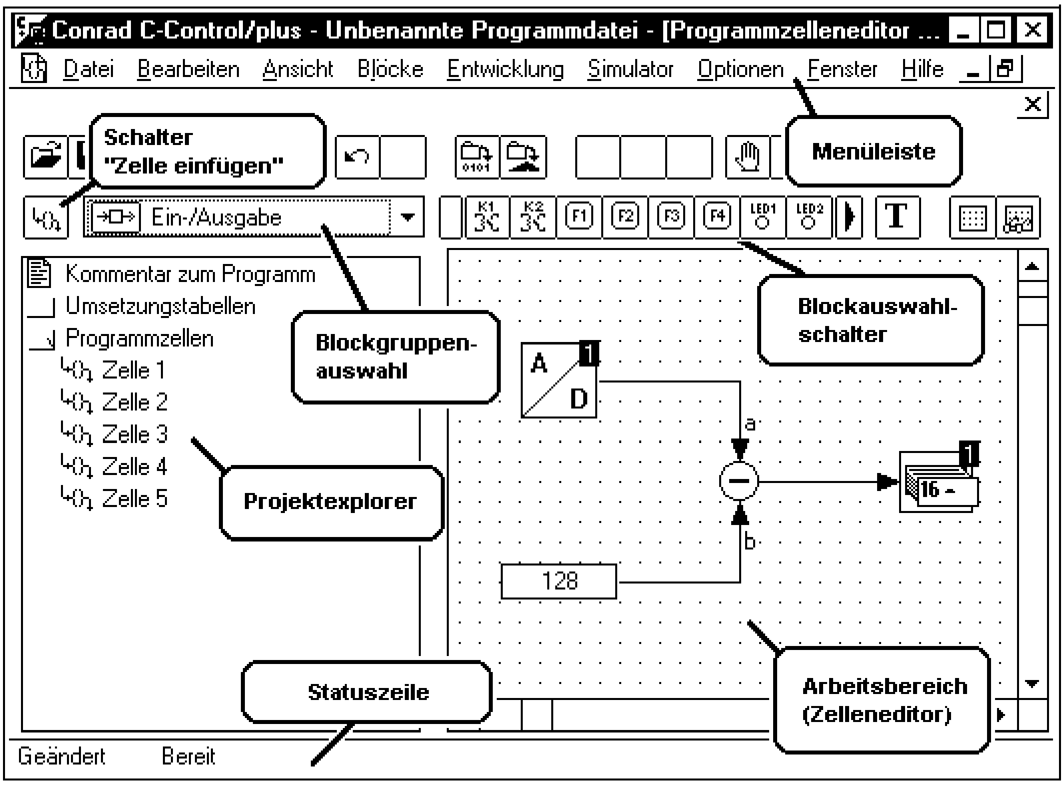


Abbildung 5

Ein grafisches Programm besteht aus mehreren sogenannten Zellen, die die einzelnen Arbeitsschritte beschreiben, welche die *C-Control Station* im Run-Modus ausführen soll. Einen Überblick über alle Zellen eines Projektes bietet der Projektexplorer im linken Bereich. Neue Zellen können dem Projekt zum Beispiel durch Klicken des Schalters „Zelle einfügen“ hinzugefügt werden.

Die Arbeitsschritte in den Zellen werden durch Blockschaltbilder festgelegt. Das heißt, einzelne Funktionsblöcke werden durch Signalflußlinien verbunden. Die Blockschaltbilder werden im Zelleneditor erstellt und bearbeitet.

Blockvorrat

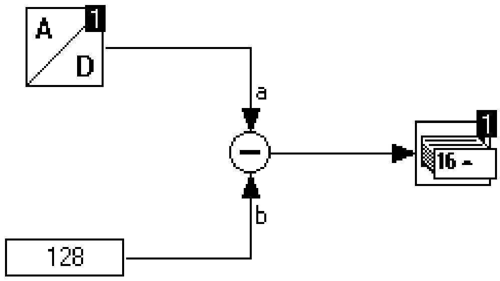


Abbildung 6

Wie Abbildung 6 zeigt, gibt es Funktionsblöcke, die Werte liefern, andere verarbeiten Werte zu einem Ergebnis und wieder andere speichern Werte oder lösen von den eingespeisten Werten abhängige Aktionen aus. Im Beispiel wird ein Meßwert vom A/D-Port 1 erfaßt, davon die Konstante 128 abgezogen (a-b) und das Ergebnis in einem Speicher abgelegt.

Der Vorrat aller in C-Control/plus zur Verfügung stehenden Funktionsblöcke unterteilt sich in folgende Gruppen:

- **Ein-/Ausgabe** - Zugriffe auf Relais,Tasten, LEDs und die Tonausgabe
- **Meßwerte** - Zugriffe auf die Temperatursensoren, die Spannungsmessung und die allgemeinen Analogports
- **Digitalports** - Zugriffe auf die digitalen Portklemmen P1 bis P6
- **Berechnungen** - mathematische Verknüpfungen und Funktionen
- **Vergleich** - Vergleichsoperationen, die im Ergebnis wahr (= -1) oder falsch (=0) sein können
- **Logik** - bitweise Verknüpfungen nach boolscher Algebra
- **Zeit** - Zugriff auf die interne Echtzeituhr (DCF77) und den Timer
- **Variablen** - Speicherblöcke
- **Konstanten** - Blöcke, die konstante Werte liefern
- **Datenaufzeichnung** - Aufzeichnung von Daten im EEPROM-Speicherchip
- **Serielle Schnittstelle** - Ein- und Ausgabe von Text und Daten über die serielle Schnittstelle des Systems

- **Programmsteuerung** - Funktionsblöcke zur Steuerung des Programmflusses

Beispiel zur grafischen Programmierung

Erstellen des Programms

Das folgende kleine Beispiel soll das Grundprinzip der grafischen Programmierung verdeutlichen. Die hier vermittelten Informationen versetzen Sie in die Lage, die zusätzlichen Beispiele von der CD nachzuvollziehen und Ihren Vorstellungen anzupassen, um so schrittweise einen vollständigen Einblick in die grafische Programmierung der *C-Control Station* zu bekommen.

Angenommen, Sie wollen mit der Taste F1 am Modul das Relais K1 abwechselnd ein- und ausschalten.

Starten Sie das Programm CCSTAT.EXE. Beginnen Sie ein neues Projekt mit dem Menübefehl „Datei/Neu“. Aktivieren Sie im Dialog „Optionen/Umgebungseinstellungen/Projektübersichtsfenster“ die Punkte „Baum der Programmzellen aufklappen“, „Erste Programmzelle öffnen“ und „Editor folgt im Projektfenster ausgewählter Programmzelle“. Diese Einstellungen werden beim Beenden des Programms gespeichert und müssen nicht immer neu eingegeben werden.

Klicken Sie im Projektexplorer auf den Ordner „Programmzellen“. Klicken Sie 6 mal auf den Schalter „Zelle einfügen“. Sie sehen, wie im Projektexplorer die neuen Zellen eingetragen werden.



Abbildung 7

Klicken Sie im Projektexplorer auf den Eintrag „Zelle 1“. In der Titelzeile des Zelleneditors steht nun „Programmzelleneditor - Zelle 1“. Klicken Sie in den Arbeitsbereich des Zelleneditors. Nun werden die bisher grauen Blockauswahlschalter aktiv.

In der Blockgruppenauswahl sollte „Ein-/Ausgabe“ stehen. Klicken Sie jetzt auf den dritten Blockauswahlschalter von links, er symbolisiert die Modultaste F1.

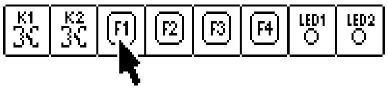


Abbildung 8

Ein neuer Block wird in den Arbeitsbereich eingefügt. Ziehen Sie den Block mit der Maus etwas nach rechts unten.

Wechseln Sie jetzt in der Blockgruppenauswahl zur Gruppe „Programmsteuerung“, sie befindet sich ganz unten in der Liste.

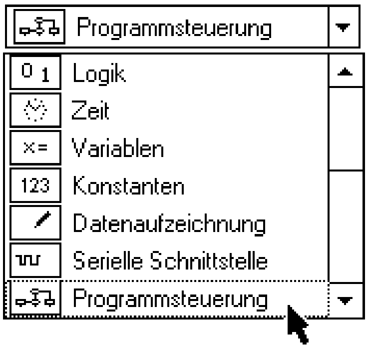


Abbildung 9

Im Bereich der Blockauswahlschalter erscheint die Palette der Programmsteuerungsblöcke. Klicken Sie auf den zweiten Schalter von links, „Unterprogrammaufruf“. Plazieren Sie den neuen Block etwas rechts neben dem ersten.

Jetzt sollen die Blöcke verbunden werden. Klicken Sie dazu auf den „F1“-Block, bewegen Sie dann den Mauszeiger in Richtung des anderen Blocks. Eine gestrichelte Linie („Gummiband“) zeigt den Verbindungsmodus an. Klicken Sie jetzt auf den zweiten Block. Im Arbeitsbereich sollte jetzt folgendes Blockschaltbild zu sehen sein:

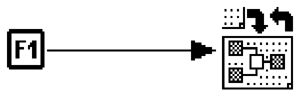


Abbildung 10

Was bedeutet das? Der Unterfunktionsaufruf des rechten Blocks ist an die Bedingung geknüpft, daß die Taste F1 am Modul gedrückt ist, „...wenn Taste F1, dann verzweige...“. Bevor wir jetzt festlegen, wohin verzweigt werden soll, wollen wir die anderen Zellen auffüllen.

Klicken Sie auf Zelle 2 im Projektexplorer. Im Arbeitsbereich plazieren Sie den ersten Block aus der Gruppe „Programmsteuerung“, den Block „Verzweige zu Zelle“.

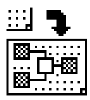


Abbildung 11

Hiermit soll die Endlosschleife des Programms realisiert werden. Aus Zelle 2 soll immer zurück zu Zelle 1 gesprungen werden. Klicken Sie mit der rechten Maustaste auf den Block. Es erscheint ein Menü. An erster Stelle finden Sie den Menüpunkt „Eigenschaften“. Wählen Sie diesen, und es öffnet sich der Eigenschaftsdialog für den Block.

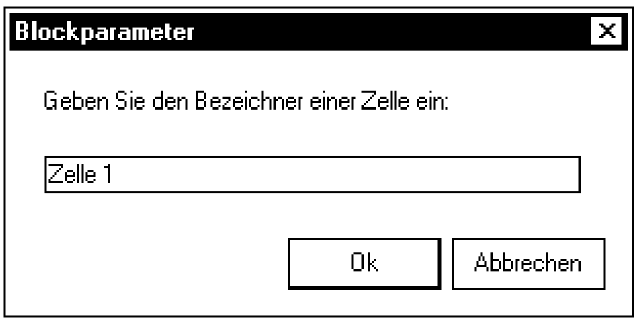


Abbildung 12

Tragen Sie in der Eingabezeile die Bezeichnung des Sprungziels des Verzweigungsblockes ein, hier Zelle 1. Achten Sie darauf, daß die Bezeichnung genau mit der im Projektexplorer übereinstimmt: großes Z, ein Leerzeichen vor der 1. Groß- und Kleinschreibung wird von der Software genauso unterschieden, wie ein, zwei oder kein Leerzeichen.

Wenn Sie falscherweise „zelle1“ eingeben, wird der Compiler einen Fehler melden!

Beenden Sie den Dialog mit Ok.

Klicken Sie jetzt im Projektexplorer auf Zelle 3. Wenn die Zelle markiert ist, klicken Sie ein zweites Mal auf Zelle 3. Es erscheint ein Eingabefeld, mit dem Sie den Zellennamen ändern können.

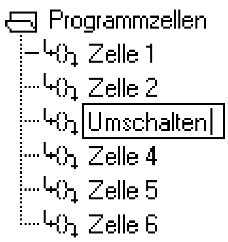


Abbildung 13

Geben Sie „Umschalten“ ein, und drücken Sie zur Bestätigung die ENTER-Taste.

Wechseln Sie jetzt wieder in den Arbeitsbereich. Plazieren Sie aus der Gruppe „Ein-/Ausgabe“ zweimal den Block „Relais K1“. Setzen Sie die Blöcke auf gleicher Höhe ein gutes Stück auseinander. Plazieren Sie dazwischen einen „Invertieren“-Block aus der Gruppe „Logik“. Verbinden Sie die Blöcke folgendermaßen:

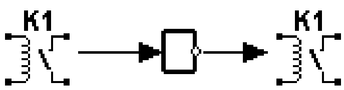


Abbildung 14

Der Sinn dieser Struktur sollte sofort einleuchten: Der Schaltzustand des Relais K1 wird gelesen und invertiert wieder ausgegeben.

Zu Zelle 4: Klicken Sie im Projektexplorer auf Zelle 4. Hier soll die Entprellung der Taste F1 untergebracht werden. Prellen nennt man die unerwünschte Eigenschaft mechanischer Kontakte, sich bis zur endgültigen Herstellung der elektrischen Verbindung in Bruchteilen einer Sekunde mehrmals zu öffnen und zu schließen. Das würde in unserem Beispiel zu einem unkontrollierten Umschalten des Relais führen.

Zwar zeigt die Folientastatur der *C-Control Station* ein geringes Prellverhalten, aber sicher ist sicher.

Plazieren Sie aus der Gruppe „Programmsteuerung“ einen „Pause“-Block. Schieben Sie ihn etwas nach rechts. Fügen Sie links neben dem „Pause“-Block einen Block „konstanter Wert“ aus der Gruppe „Konstanten“ ein. Verbinden Sie beide Blöcke. Klicken Sie mit der rechten Maustaste auf den Konstantenblock. Geben Sie über den Eigenschaftsdialog den Wert 5 ein.

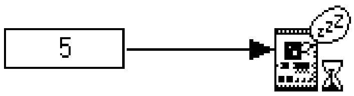


Abbildung 15

Dem „Pause“-Block wird der konstante Wert 5 eingespeist. Bei der Abarbeitung der Anweisung legt das Programm eine Pause von 5 x 20ms = 100 ms ein. In dieser Zeit erfolgt keine weitere Abfrage des Tasters, der Taster ist somit entprellt.

Zelle 5: Da wir nicht wollen, daß bei längerem Gedrückthalten der Taste F1 das Relais mehrmals umgeschaltet wird, muß das Programm nun warten, bis die Taste wieder losgelassen wird.

Dazu plazieren Sie im Arbeitsbereich der Zelle 5 aus der Gruppe „Programmsteuerung“ einen „Warten auf Wert“-Block und schieben ihn etwas nach rechts. Plazieren Sie nun links neben dem „Warten...“-Block den F1-Tasten-Block und in der Mitte einen Inverter. Beide Blocktypen sind ja inzwischen bekannt. Verbinden Sie die drei Blöcke in Reihe.

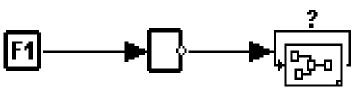


Abbildung 16

Der „Warten...“-Block verzögert die Programmabarbeitung solange, bis der an seinem Eingang eingespeiste Wert ungleich 0 ist. Da hier der invertierte Zustand der Taste F1 zugeführt wird, wartet das Programm also solange, bis die Taste losgelassen wurde. Damit wird verhindert, daß ein längeres Drücken der Taste mehrmals zum Umschalten führt.

Zelle 6 beendet die Reaktion auf einen F1-Tastendruck. Dazu fügen Sie lediglich einen „Rückkehr aus Unterprogramm“-Block aus der Gruppe

„Programmsteuerung“ ein. Nach Zelle 6 wird das Programm in der Zelle nach dem Unterprogrammaufruf fortgesetzt, also in Zelle 2.

Abschließend sei nicht vergessen, dem Block zum Unterprogrammaufruf in Zelle 1 noch das Sprungziel anzugeben. Klicken Sie auf Zelle 1 im Projektexplorer. Öffnen Sie dann den Eigenschaftsdialog des Verzweigungsblockes (zur Erinnerung: rechte Maustaste auf den Block, „Eigenschaften“ im Menü auswählen) und geben Sie als Sprungziel „Umschalten“ ein, den neuen Namen der ehemaligen Zelle 3.

Das Beispielprogramm ist jetzt vollständig. Speichern Sie das Projekt ab (Menü „Datei/Speichern“), z.B. als Datei TEST1.CPF.

Compilieren und Simulation

Das vollständige Projekt können Sie jetzt compilieren, das heißt übersetzen in ein dem C-Control System verständliches Format. Wählen Sie dazu den Menübefehl „Entwicklung/Compilieren“, oder Klicken Sie den „Compilieren“-Schalter.

Ein Fenster der Benutzeroberfläche informiert Sie nun über den Fortschritt des Compilierens und zeigt am Ende die Anzahl der erzeugten Codebytes an, oder aber eine Fehlermeldung, wenn Sie etwas falsch eingegeben haben.

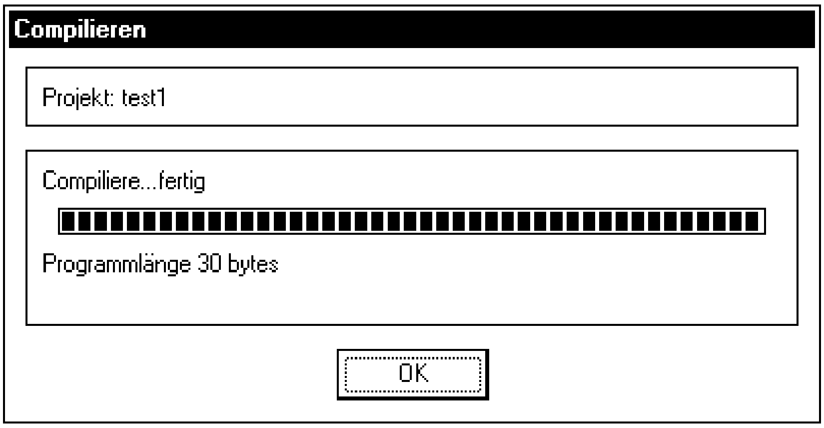


Abbildung 17

Das Ergebnis der Übersetzung bekommen Sie nicht unmittelbar zu sehen, es steht im Arbeitsspeicher Ihres PCs und kann jetzt zum Modul übertragen oder zur Simulation benutzt werden. Wenn Sie die Anzahl der Codebytes lesen, erinnern Sie sich vielleicht daran, daß insgesamt ca.

8000 Bytes zur Verfügung stehen. Sie können nun etwa abschätzen, wie komplex Ihre Steuerungsaufgaben sein können!

Die Funktion des obigen Beispielprogramms soll nun mit dem Simulator getestet werden. Öffnen Sie zunächst über das Menü „Ansicht" die Simulatorfenster „Digitalports/Relais" und „Tasten/LEDs".

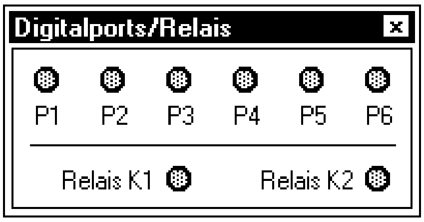


Abbildung 18

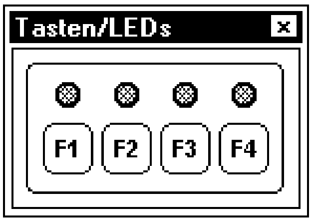


Abbildung 19

Starten Sie nun den Simulator über den Menübefehl „Simulator/Ausführen" oder den entsprechenden Toolbar-Schalter. Achten Sie auf die Anzeige „Simulation läuft" in der Statuszeile.

Klicken Sie jetzt mehrmals mit der Maus in das F1-Feld des Fensters „Tasten/LEDs". Beobachten Sie dabei die grüne Leuchtdiode K1 im Fenster „Digitalports/Relais". Die Simulatorfenster bilden die Peripheriefunktionen der C-Control Station grafisch nach. Im Simulator verhält sich das Programm wie beabsichtigt. Die Funktion des Beispiels ist also nachgewiesen. Sie können jetzt den Simulator anhalten und rücksetzen. Beide Befehle finden Sie im „Simulator"-Menü.

Der Simulator ermöglicht das Testen von Anwenderprogrammen ohne Hardware. Falls Fehler in dessen Ablauf festgestellt werden, können diese mit ein paar Mausklicks beseitigt werden. Erst das fertige Programm wird in die C-Control Station übertragen. Somit spart man viel Zeit und umständliches Handtieren mit dem Modul. Außerdem wäre ein

fehlerhaftes Programm eventuell mit unangenehmen Folgen verbunden, immerhin können die Relais auch Stromkreise mit 230V~-Netzspannung schalten!

Übertragen und Starten des Programms

Nach dem Test des Programms kann dieses übertragen werden.

- Stellen Sie sicher, daß die *Station* in zulässiger Weise mit Betriebsspannung versorgt und über eine serielle Schnittstelle mit dem PC verbunden ist.
- Drücken Sie den Reset-Taster an der *Station*.
- Stellen Sie über „Optionen / Umgebung / Simulator und Lader“ die verwendete serielle Schnittstelle ein.
- Wählen Sie den Menübefehl „Entwicklung / In C-Control-Station übertragen“.

Am Bildschirm wird der Fortschritt der Programmübertragung mit einem Balken angezeigt. Während der Übertragung blinkt die TX-Leuchtdiode an den RS232-Klemmen der Station.

Die *C-Control Station* ist jetzt programmiert.

- Drücken Sie den Start-Taster, um das kleine Testprogramm auszuführen.

Sie sehen, daß - wie programmiert und simuliert - mit jedem Drücken auf die F1-Taste am Modul das Relais K1 ein- und ausgeschaltet wird. Die Rückmeldung über den Schaltzustand des Relais vermittelt die Leuchtdiode an der Relais-Klemme.

Mit den Erfahrungen aus diesem kleinen Testprogramm und den weiteren Beispielen von der CD sind Sie in der Lage, schrittweise komplexere Anwendungen zu realisieren.

Programmieren in BASIC

Überblick

Zur **Realisierung komplexer Projekte** oder für Anwender mit entsprechenden Vorkenntnissen empfiehlt sich die Programmierung der *C-Control Station* in der **Programmiersprache BASIC** , in C-Control/BASIC (CCBASIC). Dazu finden Sie zwei Versionen der Programmieroberfläche auf der der *Station* beiliegenden CD. Die Hauptversion benötigt als Betriebssystem mindestens Microsoft Windows95 oder WindowsNT ab 4.0 oder entsprechende Nachfolgeversionen! Die Hauptversion von C-Control/BASIC kann nicht unter dem inzwischen veralteten Windows 3.xx installiert werden.

Auf der CD befindet sich jedoch auch eine abgerüstete Version (ohne Simulator). Diese ist lauffähig unter MS-DOS und läßt sich somit auch auf den meisten älteren PCs und Notebooks betreiben, bis hinab zu 8086/88-Systemen (XT). Preiswerte ältere Notebooks, die heutiger Bürosoftware nicht mehr gewachsen sind, eignen sich so hervorragend zur Umprogrammierung einer *C-Control Station* vor Ort und ohne Ausbau des Moduls aus dem Schaltschrank.

Zur Installation beider Versionen befolgen Sie bitte wieder die Hinweise auf der CD.

Grundlegendes zur Programmiersprache CCBASIC

Allgemeines

Ein BASIC-Programm besteht aus mehreren Zeilen sogenannten Quelltextes. Dabei enthält jede Zeile eine oder mehrere Rechen- oder Steueranweisungen. CCBASIC ist der BASIC-Dialekt, der zur Programmierung der *C-Control Station* verwendet wird. Die Syntax entspricht in etwa der des Standard-BASIC. Bei einigen Befehlen gibt es Abweichungen oder Erweiterungen, die speziell auf die Hardware der *C-Control Station* zugeschnitten sind.

Enthält eine Programmzeile mehrere Anweisungen, dann sind diese durch Doppelpunkte : zu trennen.

Zeilennummern, wie in älteren BASIC-Dialekten üblich, sind nicht notwendig. Werden dennoch Zeilennummern angegeben, so können diese als Sprungziel verwendet werden.

```
10 ...
GOTO 10
```

Einen Einfluß auf die Reihenfolge der Programmoperationen haben die Nummern darüber hinaus nicht. Wenn beispielsweise im Quelltext auf eine mit 200 numerierte Zeile eine Zeile 100 folgt, wird dennoch die Zeile 200 vor der 100 abgearbeitet.

Kommentare können zur Erläuterung des geschriebenen Programms mit in den Quelltext aufgenommen werden und steigern dessen Lesbarkeit und Wartungsfreundlichkeit. Ein Kommentar in CCBASIC beginnt stets mit einem Hochkomma **'** und erklärt den Rest der Zeile zum nicht zum Programm gehörigen Text.

```
a = b + c    '... Kommentar ...
```

Bezeichner

Bezeichner sind Programmelemente aus alphanumerischen Zeichen (A bis Z, 0 bis 9) die in vom Programmierer festgelegter Weise Objekte, wie Variablen und Konstanten, bezeichnen. Label-Namen und die sogenannten „reservierten Worte“ sind ebenfalls Bezeichner.

Es erfolgt keine Unterscheidung von Groß- und Kleinbuchstaben. Ein Bezeichner beginnt stets mit einem Buchstaben oder mit einem Unterstrich. Leerzeichen innerhalb eines Bezeichners sind nicht erlaubt.

Variablen und Konstanten

Variablen und Konstanten sind Objekte des Informationsverarbeitungs-prozesses. In CCBASIC speichern beide einen numerischen Wert. Während der Wert einer Konstante einmal angegeben wird und dann unverändert bleibt, kann sich der Wert einer Variablen im Lauf des Programms beliebig oft ändern.

Konstanten können in CCBASIC in dezimaler, hexadezimaler und binärer Form angegeben werden. Die Syntax für Hexadezimal- und Binärzahlen sei hier am Beispiel der Zahl 46 (dezimal) gezeigt:

```
&H2E        ' hexadezimal
&B101110    ' binaer
```

Auf Variablen wird stets über ihren Bezeichner zugegriffen. Dieser Bezeichner muß vor der ersten Verwendung der Variable im Programm in einer DEFINE-Zeile definiert werden.

Label

Label markieren bestimmte Punkte in der Folge der Programmoperationen. Label sind Ziele von Sprungoperationen innerhalb eines Algorithmus. In CCBASIC stehen Label am Anfang einer Zeile und beginnen stets mit einem Doppelkreuz, dann folgt - ohne Leerzeichen - der Bezeichner des Labels.

Das Beispiel zeigt die Definition des Labels „Label1“ und die Verwendung in einem Sprungbefehl:

```
#label1 ...
GOTO label1
```

Terme

Ein Term ergibt sofort (als Variable oder Konstante) oder durch Berechnung einen bestimmten Wert. Terme sind Teile von Anweisungen und stehen beispielsweise bei der Zuweisung eines Wertes an eine Variable rechts des Zuweisungszeichens „=“. Terme werden durch Kombinationen von Operanden und Operatoren gebildet.

```
100
c
a + b
(ABS(x) - 13) * 10
```

Operanden und Operatoren

Ein Operand ist in der Grundform entweder eine Konstante, eine Variable oder ein Funktionsaufruf, kann aber auch selbst wieder ein aus Operanden und Operatoren zusammengesetzter Term sein.

Operatoren bezeichnen Rechenoperationen, die mit den umstehenden Operanden auszuführen sind. Dabei gibt es eine definierte Rangfolge der Operatoren (siehe Befehlsbeschreibung), die die Reihenfolge der Berechnungen bestimmt.

Funktionen

Eine Funktion führt eine definierte Operation - zum Beispiel eine Berechnung - durch und liefert durch ihren Aufruf einen Ergebniswert. Die meisten Funktionen erwarten ein oder mehrere Argumente, die in runden Klammern nach dem Funktionsbezeichner „( )“ übergeben werden und durch Kommas getrennt sind. Einige Funktionen werden ohne Argument aufgerufen. In diesem Fall werden keine runden Klammern geschrieben.

```
ABS(x)
MAX(a,b)
RAND
EOF
```

In CCBASIC sind alle unterstützten Funktionen vordefiniert. Deren Bezeichner gehören zu den reservierten Worten.

Zuweisungen

Die Zuweisung ist die einfachste Form einer Programmanweisung. Nach dem Bezeichner einer Variablen, die einen Wert zugewiesen bekommen soll, folgt das Zuweisungszeichen „=“ und dann ein Term, der den zugewiesenen Wert bestimmt. Eine Zuweisung entspricht damit einer einfachen mathematischen Formel.

```
a = 10
b = x - y
c = SQR(a*a + b*b)
```

Befehle

Neben den einfachen Zuweisungen sind Befehle Anweisungen zur Ausführung von Programmoperationen durch die *C-Control Station*. Befehle beginnen stets mit einem reservierten Wort. Einige Befehle erwarten einen oder mehrere Parameter zur genauen Spezifikation der auszuführenden Programmoperation. Diese Parameter werden nach dem Befehlsbezeichner und einem Leerzeichen aufgeführt und dabei durch Kommas getrennt (Ausnahme PRINT, siehe Befehlsübersicht). Im Gegensatz zu den Argumenten beim Aufruf einer Funktion stehen die Befehlsparameter nicht innerhalb runder Klammern!

```
RANDOMIZE
PAUSE 100
BEEP 440,50,50
```

Anweisungen zur Steuerung des Programmflusses

Diese Anweisungen erlauben, die Reihenfolge der an sich streng sequentiell abgearbeiteten Programmoperationen zu steuern und an Eingangswerte des Informationsverarbeitungsprozesses anzupassen. Sie bieten eine hohe Flexibilität bei der Algorithmenformulierung und sind für die Lösung mancher anwendungstechnischer Probleme sogar Grundvoraussetzung.

Anweisungen zur Steuerung des Programmflusses bestehen aus einem oder mehreren reservierten Worten und erfordern in jeweils spezieller Weise eventuell weitere Angaben.

```
GOTO label1
IF a > b THEN GOSUB label2

FOR i = 0 TO 10 STEP 2
...
NEXT
```

Compileranweisungen

Zusätzlich zu den Programmanweisungen enthält ein CCBASIC-Quelltext Compileranweisungen, die zum Beispiel zum Anlegen von Datenblocks (Tabellen) oder zur Definition von Variablen- und Konstanten dienen.

Für Compileranweisungen gilt die Doppelpunktregel zum Trennen mehrerer Anweisungen in einer Zeile nicht. Es darf jeweils nur eine Compileranweisung in einer Zeile stehen.

Die DEFINE-Anweisung

Die **DEFINE**-Anweisung ist eine Compileranweisung.

Definition symbolischer Konstanten

Es ist guter Programmierstil, statt „magischer“ Zahlen im Programm

```
IF x > 1234 THEN GOTO alarm
```

besser symbolische Konstanten zu verwenden. Durch Vergabe signifikanter Bezeichner für Konstanten erhöht sich die Lesbarkeit des Quelltextes. Wenn alle Konstanten global definiert werden, ist ein Programm auch leichter zu warten. Das gilt besonders, wenn ein und dieselbe Konstante mehrmals im Programm benötigt wird.

Die Definition einer symbolischen Konstante erfolgt in der Art:

```
DEFINE bezeichner wert
```

Dabei ist wert entweder eine dezimale, hexadezimale oder binäre Zahl.

So sollte das Beispiel oben besser

```
DEFINE limit 1234
...
IF x > limit THEN GOTO alarm
```

lauten.

Definition von Variablen

Das Betriebssystem der *C-Control Station* stellt 24 Byte-Speicherzellen des internen Speichers (RAM) dem Anwender zur Verwendung in seinen Programmen zur Verfügung. In diesem Speicherbereich werden alle Variablen eines BASIC-Programms gespeichert. Die 24 Bytes können je nach Bedarf auch bitweise oder als 16bit Integer (Word) verwendet werden.

Im Gegensatz zum Standard-BASIC müssen in CCBASIC alle vom Programm benutzten Variablen vor ihrer ersten Verwendung definiert werden. Dabei ist der Datentyp zu spezifizieren (Bit, Byte oder Word) und kann (für Bits muß!) eine Speicherzellennummer angegeben werden. Der Anwender muß selbst darauf achten, daß keine unerwünschten Überlappungen bei der Vergabe der Speicherplätze entstehen, da es sonst zum gegenseitigen Überschreiben der Variablen kommen kann. Beispielsweise belegen bit[18], byte[2] und word[1] jeweils einen Teil der Zelle 2 des Speicherbereiches.

- Definition einer Bitvariablen:

```
DEFINE bezeichner BIT[nr]
```

Dabei sind für nr Werte von 1 bis 192 (24 Bytes mit je 8 Bit) zulässig.

- Definition einer Bytevariablen mit Zellennummer:

```
DEFINE bezeichner BYTE[nr]
```

Dabei sind für nr Werte von 1 bis 24 (24 Bytes) zulässig.

- Definition einer Integervariablen mit Zellennummer:

```
DEFINE bezeichner WORD[nr]
```

Dabei sind für nr Werte von 1 bis 12 (ein Word belegt 2 Bytes) zulässig.

Wenn bei Byte- und Worddefinitionen die Zellenangabe [nr] weggelassen wird, übernimmt der Compiler die Aufteilung auf den Speicherbereich. Achten Sie dann darauf, daß nicht abwechselnd Bytes und Words definiert werden. Die folgenden Anweisungen

```
DEFINE a BYTE
DEFINE b WORD
DEFINE c BYTE
DEFINE d WORD
```

führen zu zwei ungenutzten (verschenkten kostbaren!) Bytes, zwischen a und b sowie zwischen c und d, da Words prinzipiell an den Bytes 1,3,5,7,... usw. der 24 Bytes ausgerichtet werden.

Besser wäre,

```
DEFINE b WORD
DEFINE d WORD
DEFINE a BYTE
DEFINE c BYTE
```

zu schreiben.

Die automatische Aufteilung der Variablen auf den Speicher durch den Compiler beginnt bei Zellennummer 1. Das obige (bessere) Beispiel belegt 6 Bytes. Bei Definition weiterer Bits, Bytes und Words mit Angabe der Zellennummer ist wieder auf unerwünschte Überlappung zu achten.

Ein bereits definierter Variablenbezeichner darf nicht ein zweites Mal definiert werden.

Definition von Digitalports

In CCBASIC wird auf Ports wie auf Variablen zugegriffen. Auch hier muß jeder verwendete Port zuvor definiert sein. Insgesamt sind 16 Ports ansprechbar.

- Definition eines der 16 Digitalports:

```
DEFINE bezeichner PORT[nr]
```

Dabei sind für nr Werte von 1 bis 16 zulässig.

Wie im Abschnitt Peripherie beschrieben, sind die Ports fest mit verschiedenen Systemkomponenten verbunden. So bietet sich meist folgender Definitionsblock an:

```
DEFINE P1 PORT[1]
DEFINE P2 PORT[2]
```

```
DEFINE P3 PORT[3]
DEFINE P4 PORT[4]
DEFINE P5 PORT[5]
DEFINE P6 PORT[6]

DEFINE K1 PORT[7]
DEFINE K2 PORT[8]

DEFINE F1 PORT[9]
DEFINE F2 PORT[10]
DEFINE F3 PORT[11]
DEFINE F4 PORT[12]

DEFINE LED1 PORT[13]
DEFINE LED2 PORT[14]
DEFINE LED3 PORT[15]
DEFINE LED4 PORT[16]
```

Definition von Analogports

```
DEFINE bezeichner AD[nr]
```

Dabei sind für nr Werte von 1 bis 8 zulässig.

Auch die Analogports sind im System fest aufgeteilt. Die übliche Definition sollte lauten:

```
DEFINE A1 AD[1]
DEFINE A2 AD[2]
DEFINE A3 AD[3]
DEFINE A4 AD[4]

DEFINE T1 AD[5]
DEFINE T2 AD[6]

DEFINE U1 AD[7]
DEFINE U2 AD[8]
```

Befehlsübersicht

Dieses Kapitel gibt einen kompletten Überblick über die CCBASIC Operatoren, Funktionen und Anweisungen.

Mathematische und logische Operatoren

- Grundrechenarten: **+ - * /**

- Der Modulooperator **MOD** liefert den Rest einer Integerdivision,

```
a = 10 MOD 3
```

ergibt beispielsweise für a den Wert 1.

- Vergleichsoperatoren: **>** (größer als), **<** (kleiner als), **>=** (größer oder gleich), **<=** (kleiner oder gleich), **=** (gleich), **<>** (ungleich)

Das Ergebnis einer Vergleichsoperation ist entweder -1 (nicht 1!, Vergleich wahr) oder 0 (Vergleich falsch).

```
a = 10 < 3
```

ergibt beispielsweise für a den Wert 0.

- logische Operatoren: **NOT** (Negation), **AND** (Und-Verknüpfung), **NAND** (Und-Verknüpfung mit anschließender Negation), **OR** (Oder-Verknüpfung), **NOR** (Oder-Verknüpfung mit anschließender Negation), **XOR** (Exklusiv-Oder-Verknüpfung)

Die logischen Operatoren können außer zur Formulierung von Bedingungen (meist in Verbindung mit Vergleichsoperationen) auch für binäre Byte- oder Wordmanipulationen benutzt werden.

- Schiebeoperatoren: **SHL** (nach links schieben), **SHR** (nach rechts schieben) werden zum arithmetischen Verschieben von Bitmustern in Byte- oder Wordvariablen benutzt. Links des Operators steht der zu schiebende Wert, rechts die Zahl, um wieviel Bits verschoben werden soll. Beim Linksschieben entspricht jede einzelne Verschiebung einer Multiplikation mit 2, beim Rechtsschieben einer Division durch 2.

```
a = 10 SHL 3
```

entspricht also

```
a = 10 * 2 * 2 * 2
```

und ergibt beispielsweise für a den Wert 80.

Achten Sie darauf, daß SHR arithmetisch schiebt, also mit Vorzeichen!

```
a = &B1000000000000000 SHR 1
```

ergibt (alle Werte in Binärnotation) für a den Wert &B1100000000000000 nicht &B0100000000000000!

Mathematische Funktionen und Befehle

Die Argumente x und y, je nach Funktion oder Befehl, sind stets Terme (Definition siehe oben).

- Die Wurzelfunktion **SQR(x)** liefert eine Näherung für die Quadratwurzel aus dem Argument x. Dabei werden die Nachkommastellen abgeschnitten.
- Die Signumfunktion **SGN(x)** ergibt 1, wenn der Wert des Argumentes x größer 0, und ergibt -1, wenn der Wert kleiner 0 ist. Für x = 0 ist auch das Ergebnis der SGN-Funktion gleich 0.
- Die Maximumfunktion **MAX(x,y)** ergibt x, wenn x > y ist, sonst y.
- Die Minimumfunktion **MIN(x,y)** ergibt x, wenn x < y ist, sonst y.
- Der Befehl **RANDOMIZE x** initialisiert den internen Pseudo-Zufallsgenerator des Steuercomputers mit dem Wert von x. Ein und derselbe Initialisierungswert führt stets zu einer identischen Folge von Zahlen. Die Spezialform **RANDOMIZE TIMER** lädt den Wert des freilaufenden Timers in den Generator.
- Die Zufallsfunktion **RAND** liefert den nächsten Integer-Zufallswert des Pseudo-Zufallsgenerators. Die Zufallszahlen werden nach dem multiplikativen Verfahren mit anschließender Modulodivision aus dem jeweils vorangehenden Wert erzeugt.

Rangfolge von Operatoren und Funktionsaufrufen

Bei der Berechnung von Termen mit Operatoren und Funktionen ist deren Rangfolge von entscheidender Bedeutung. Teilausdrücke mit Operatoren von hohem Rang werden vor denen mit einem niedrigerem Rang berechnet (vergleiche Rechenregel: „Punktrechnung vor Strichrechnung“). Bei gleichrangigen Operatoren erfolgt die Berechnung von links nach rechts.

Wie in der Mathematik üblich kann jedoch durch Klammersetzung zusätzlich Einfluß auf die Berechnungsreihenfolge genommen werden. CCBASIC unterstützt maximal 6 Klammerebenen.

Im Sinne der Übersichtlichkeit eines Programmes sollten jedoch „wilde“ Klammerausdrücke vermieden und komplexe Berechnungen auf mehrere BASIC-Zeilen aufgeteilt werden.

Die folgende Liste zeigt die CCBASIC Operatorenrangfolge :

Rang	Operatoren
9	( )
8	Funktionsaufrufe
7	negatives Vorzeichen
6	* / MOD SHL SHR
5	+ -
4	> >= < <= = <>
3	NOT
2	AND NAND
1	OR NOR XOR

Anweisungen zur Steuerung des Programmflusses

- Schleife

```
FOR variable = anfang TO ende STEP schrittweite
...
NEXT
```

Die FOR-Schleife führt die Anweisungen bis zum NEXT solange aus, bis der Wert der variable gleich dem Wert des Terms ende ist. Vor dem ersten Durchlauf wird der Wert des Terms anfang berechnet und der Schleifenvariablen zugewiesen. In jedem Durchgang wird der Wert des schrittweite-Terms zur Schleifenvariablen addiert. In der Form

```
FOR variable = anfang TO ende
...
NEXT
```

beträgt die Schrittweite konstant 1.

Die Werte des ende-Terms und des schrittweite-Terms werden mit jedem Schleifendurchlauf neu berechnet. Das gestattet eine erweiterte Kontrolle des Programmverlaufes.

FOR-Schleifen können ineinander verschachtelt werden. Die Verschachtelungstiefe ist nur durch den für die Schleifenvariablen erforderlichen Speicherplatz beschränkt.

```
FOR v1 = anfang1 TO ende1
```

```
  FOR v2 = anfang2 TO ende2
    FOR v3 = anfang3 TO ende3
    ...
    NEXT
  NEXT
NEXT
```

Jede FOR-Schleife darf im Verlauf des Programms nur über ihre eigene NEXT-Anweisung laufen. Folgender Quelltext kann zwar compiliert und in die *C-Control Station* geladen werden, wird jedoch nicht wie vielleicht erwartet funktionieren:

```
FOR v1 = anfang1 TO ende1
...
  GOTO anothernext
...
NEXT

FOR v2 = anfang2 TO ende2
...
#anothernext
NEXT
```

Achten Sie außerdem auf den Werteberech von Schleifenvariable und ende-Term!

```
DEFINE v BYTE

FOR v = 1 TO 1000
...
NEXT
```

wird zu einer Endlosschleife, da v als Bytevariable nie den Wert 1000 erreichen kann, sondern bereits nach 255 wieder auf 0 überrollt.

- Bedingte Ausführung

```
IF bedingung THEN anweisung1
```

oder

```
IF bedingung THEN anweisung1 ELSE anweisung2
```

Die IF...THEN...ELSE-Konstruktion ermöglicht die Anpassung des Programmflusses an Bedingungen zur Laufzeit des Programms. Als bedingung ist ein beliebiger Term einzusetzen. Ergibt dessen Berechnung einen Wert ungleich 0, dann gilt die Bedingung als erfüllt, und die anweisung1 wird ausgeführt. Werden zusätzlich ein ELSE und eine zweite Anweisung angegeben, so wird diese Anweisung alternativ ausgeführt, wenn der berechnete Term einen Wert gleich 0 ergibt.

Die gesamte IF...THEN...ELSE-Konstruktion muß in einer Quelltextzeile stehen. Anweisungsblöcke (mehrere Anweisungen) nach THEN und ELSE sind nicht zulässig.

- Sprunganweisung

```
GOTO label
```

Mit der GOTO-Anweisung kann der Steuercomputer veranlaßt werden, die Programmabarbeitung an einer bestimmten Stelle fortzusetzen. Als Ziel des Sprungs wird ein Label-Bezeichner angegeben. Das Sprungziel kann sich vor oder nach der GOTO-Anweisung im Quelltext befinden.

- Aufruf und Rückkehr aus einer Unterroutine

Der Aufruf einer Unterroutine erfolgt mit der Anweisung

```
GOSUB label
```

Dabei ist label der Anfangspunkt der Unterroutine.

In den sogenannten Unterroutinen sind Programmabschnitte zusammengefaßt, die mehrfach im Verlauf der Programmabarbeitung benötigt werden. Eine Unterroutine beginnt stets mit einem Label, enthält dann eine oder mehrere Anweisungen und abschließend ein

```
RETURN
```

Nach dem RETURN wird die Programmabarbeitung mit der Anweisung nach dem GOSUB fortgesetzt. Die Programmabarbeitung darf ohne ein vorheriges GOSUB niemals an eine RETURN-Anweisung gelangen.

Die maximal zulässige Verschachtelungstiefe bei Aufrufen von Unterroutinen aus Unterroutinen ist vier.

```
#hauptprogramm
  GOSUB sub1
  ...

#sub1
  GOSUB sub2
  ...
  RETURN

#sub2
  GOSUB sub3
  ...
  RETURN
```

```
#sub3
  GOSUB sub4
  ...
 RETURN

#sub4
 ...
 RETURN
```

Über RETURN kann auch ein Zahlenwert an den Aufrufer zurückgegeben werden.

```
 RETURN term
```

Somit können wiederkehrende oder umfangreichere Berechnungen oder Statusermittlungen in eine Unterfunktion gekapselt werden.

```
define x word
define y word
define z word

...
  x = calc
...

#calc RETURN y * z / (y + z)
```

- Wertgesteuerte Programmverzweigung

```
ON variable GOTO label0,label1,...labeln
```

oder

```
ON variable GOSUB label0,label1,...labeln
```

In Abhängigkeit des Wertes des Selektors variable erfolgt eine Programmverzweigung oder ein Unterroutinenaufruf zu den aufgelisteten Einsprungpunkten. Ist der Wert 0, dann wird zu label0 verzweigt, bei Wert gleich 1 zu label1 usw. Ist der Variablenwert negativ oder größer als die Anzahl der aufgeführten Sprungziele, dann wird die Programmabarbeitung ohne Verzweigung fortgesetzt.

- Programmende

```
END
```

Gelangt der Steuercomputer im Verlauf der Programmabarbeitung zur END-Anweisung, wird die Programmabarbeitung beendet. Das System verharrt dann in einem inaktiven Zustand. Jetzt kann ein neues Anwenderprogramm übertragen oder die Ausführung per Start-Taster wieder gestartet werden.

- Verzögerung des Programmflusses

Die Anweisung

```
WAIT conditionterm
```

unterbricht die Programmausführung solange, bis die Berechnung des conditionterm einen Wert ungleich 0 ergibt.

```
define F1 port[9]
...
WAIT F1
```

In diesem Beispiel wird solange gewartet, bis vom Digitalport 9 (Taste F1 an der *Station*) ein HIGH-Pegel gelesen wird.

Der PAUSE Befehl unterbricht die Programmausführung für eine gewisse Zeit. Der berechnete Wert des Parameterterms geht als Multiplikationsfaktor mit der Grundeinheit von 20 Millisekunden in die Festlegung der Pausenzeit ein.

```
PAUSE term
```

Beispielsweise wird durch den Befehl

```
PAUSE 50
```

die Programmausführung für ca. 50*20 Millisekunden = 1 Sekunde unterbrochen. Die maximale Zeitabweichung der tatsächlichen Pause vom angegebenen Wert beträgt dabei prinzipbedingt +/- 20 Millisekunden.

Kommunikation über die serielle Schnittstelle

- Datenausgabe

Die Datenausgabe erfolgt als Text über die serielle Schnittstelle der *C-Control Station*. Ist über ein Schnittstellenkabel zum Beispiel ein PC mit einem Terminalprogramm angeschlossen, können die ausgegebenen Daten dort angezeigt werden.

```
PRINT term
```

gibt das Ergebnis der Berechnung von term aus.

```
PRINT "text"
```

überträgt den in Anführungszeichen stehenden Text.

In beiden Fällen wird an die Übertragung ein Zeilenvorschubzeichen angehängt, welches das Terminalprogramm veranlaßt, die nächste Ausgabe in der nächsten Bildschirmzeile vorzunehmen. Der Zeilenvorschub kann unterdrückt werden, wenn dem PRINT-Befehl nach dem Parameter (term oder “text“) ein Semikolon hinzugefügt wird.

```
PRINT term;
```

oder

```
PRINT "text";
```

CCBASIC unterstützt außerdem mehrere Ausgaben mit einem PRINT-Befehl, wobei die einzelnen Parameter durch Komma oder Semikolon getrennt werden. Ein Komma fügt in die Ausgabe ein Tabulatorzeichen ein, das entsprechend den Einstellungen im Terminalprogramm als eine Anzahl von Leerzeichen am Bildschirm erscheint. Sollen zwei Ausgaben ohne Zwischenraum aufeinander folgen, so sind diese im PRINT-Befehl durch ein Semikolon zu trennen.

```
PRINT "a= ", a
PRINT "a= "; a
```

Ein einzelner PRINT-Befehl ohne Parameter gibt nur einen Zeilenvorschub aus.

```
PRINT
```

- Dateneingabe

Mit dem Befehl

```
INPUT variable
```

kann ein Integerwert von der seriellen Schnittstelle gelesen und für die anschließende Weiterbearbeitung in einer Variablen gespeichert werden.

Der Wert wird in einem Terminalprogramm an einem PC eingegeben und nach dem Drücken der ENTER-Taste per Schnittstellenkabel an die *C-Control Station* übertragen.

Der INPUT-Befehl wartet solange, bis eine komplette Datenübertragung vom Terminal empfangen wurde. Wird der INPUT-Befehl aufgerufen,

ohne daß eine Datenübertragung vom Terminal erfolgt, wird das Programm endlos an dieser Stelle stehen bleiben! Hier hilft dann nur noch der Reset-Taster und der anschließende Neustart der *Station*.

- Byteweise Kommunikation über die serielle Schnittstelle

Während PRINT und INPUT kurze Zeichenketten zur Darstellung eines numerischen Wertes senden beziehungsweise erwarten, kann es wünschenswert sein, einzelne Bytes seriell zu übertragen. Dafür bietet CCBASIC die Befehle PUT und GET.

```
PUT term
```

sendet den berechneten Wert eines Terms. Falls erforderlich, wird das Ergebnis zuvor auf den Byte-Wertebereich (0...255) reduziert.

```
GET variable
```

wartet auf ein seriell empfangenes Byte und speichert den Wert dann in der angegebenen Variablen.

- Weitere Schnittstellenbefehle und -funktionen

Wie beschrieben warten INPUT und GET unter Umständen endlos auf den Empfang serieller Daten. Soll ein „Aufhängen" des Programms in dieser Art verhindert werden, kann vor jedem von INPUT oder GET durch Aufruf der Statusfunktion **RXD** ermittelt werden, ob empfangene Daten zur Verfügung stehen. Die Funktion liefert in diesem Fall den Wert 1. Ist der Schnittstellenpuffer leer, so ist das Funktionsergebnis gleich 0.

```
...
if RXD then GET thebyte
...
```

Die voreingestellte Übertragungsrate der seriellen Schnittstelle beträgt für Sender und Empfänger 9600 Bit pro Sekunde (baud). Mit dem BAUD Befehl können jedoch auch andere Raten eingestellt werden. CCBASIC enthält dafür einige vordefinierte Konstaten: **R1200**, **R2400**, **R4800**, **R9600** für die Raten 1200 bis 9600 Bit pro Sekunde.

```
BAUD R2400
```

schaltet beispielsweise Sender und Empfänger auf die Rate von 2400 Bit pro Sekunde um. Prinzipiell sind auch andere als die vordefinierten Raten, auch für Sender und Empfänger unterschiedliche, möglich. Die Übertragungsraten der seriellen Schnittstelle werden durch Teilung aus

einem internen Takt des Mikroprozessors des C-Control/*BASIC* Steuercomputers abgeleitet. Der dem BAUD Befehl zu übergebende Bytewert enthält die erforderlichen Teilerwerte Nxx.

b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
NP1	NP0	NT2	NT1	NT0	NR2	NR1	NR0

Bit 7 und 6 enthalten einen für Sender und Empfänger gemeinsamen Vorteiler NP. NP kann die Werte 1, 3, 4 und 13 annehmen. Die folgende Tabelle zeigt die dafür erforderlichen Einstellungen für NP1 und NP0:

Vorteiler	**NP1**	**NP0**
1	0	0
3	0	1
4	1	0
13	1	1

NT (Bit 3 bis 5) und NR (Bit 0 bis 2) bestimmen weitere Teilerwerte, getrennt für Sender (NT) und Empfänger (NR), entsprechend folgender Codierung:

Teiler	**NT2 bzw. NR2**	**NT1 bzw. NR1**	**NT2 bzw. NR2**
1	0	0	0
2	0	0	1
4	0	1	0
8	0	1	1
16	1	0	0
32	1	0	1
64	1	1	0
128	1	1	1

Die Übertragungsrate des Senders berechnet sich nach folgender Formel

Senderrate = 125000 / (NP * NT),

die des Empfängers entsprechend

Empfängerrate = 125000 / (NP * NR).

Die weiteren Schnittstellenparameter - 8 Datenbits, kein Paritätsbit, 1 Stopbit - sind fest und können nicht geändert werden.

Dateifunktionen

Die Dateifunktionen erlauben das Aufzeichnen von Messwerten oder anderen Daten oder können zum Abspeichern von Information benutzt werden, die nach Ausfall der Betriebsspannung wieder in die Programmvariablen geladen werden sollen.

Der Speicherbereich im EEPROM-Chip nach dem Anwenderprogramm - meist der größte Teil - steht für diesen Zweck zur Verfügung. Der Speicherbereich wird als eine Datei verwaltet, auf die lesend oder schreibend zugegriffen werden kann, nachdem sie mit dem entsprechenden Attribut geöffnet wurde. Der Befehl zum Öffnen der Datei lautet wie folgt:

```
OPEN# FOR WRITE
```

oder

```
OPEN# FOR APPEND
```

oder

```
OPEN# FOR READ
```

Dabei bedeutet WRITE das Öffnen zum Schreiben mit Überschreiben eventueller alter Aufzeichnungen, APPEND das Öffnen zum Schreiben mit Anhängen der neuen an die alten Aufzeichnungen und READ das Öffnen zum Auslesen der Aufzeichnungen.

Es können nur Integerwerte gespeichert und gelesen werden. Jeder Wert belegt also 2 Bytes im EEPROM. Das Schreiben und Lesen erfolgt mit den Befehlen

```
PRINT# term
```

, wobei das berechnete Ergebnis des Terms gespeichert wird, beziehungsweise

```
INPUT# variable
```

, wobei variable eine definierte Integervariable des Programms bezeichnet.

Schreiben in und Lesen aus der Datei erfolgt streng sequentiell. Dafür wird intern ein Dateizeiger verwaltet, der nach jedem Zugriff um 1 erhöht wird.

Vor jedem Schreiben sollte geprüft werden, ob noch genügend Platz im EEPROM zur Aufnahme der Daten vorhanden ist. Dafür kann die Funktion **FILEFREE** abgefragt werden, die als Ergebnis die Größe des noch freien Speichers liefert (in Words). Folgendes Beispiel zeigt die Anwendung der Funktion

```
DEFINE a WORD
DEFINE b WORD
DEFINE c WORD
DEFINE blocksize 3
...
IF FILEFREE >= blocksize THEN GOSUB writeblock
...
#writeblock
   PRINT# a
   PRINT# b
   PRINT# c
RETURN
```

Vor jedem Lesen sollte geprüft werden, ob noch weitere Daten aufgezeichnet sind. Die Funktion dafür lautet **EOF** („end of file“). Ihr Ergebnis ist 1, wenn in der Datei keine weiteren Daten verfügbar sind, sonst 0. Die Abfrage der EOF Funktion sollte das Auslesen von Datenblocks in gleicher Weise rahmen, wie beim Schreiben der Daten. Zum obigen Beispiel würde also

```
IF NOT EOF THEN GOSUB readblock
...
#readblock
   INPUT# a
   INPUT# b
   INPUT# c
RETURN
```

gehören.

Nach Beenden eines Dateizugriffs sollte die Datei sofort wieder geschlossen werden. Erst dann sind die Daten vor einem Spannungsausfall oder Reset des Systems sicher. Der Befehl dafür lautet

```
CLOSE#
```

und hat keinen Parameter.

Portbefehle

- der Umschaltbefehl **TOG**

Prinzipiell erfolgt der Zugriff auf die Ports des Steuercomputers wie auf Variablen. Um einen Digitalport P einzuschalten, schreibt man

```
P = 1
```

und

```
P = 0
```

, um ihn auszuschalten.

Um den Port umzuschalten (EIN nach AUS; AUS nach EIN), kann man schreiben

```
P = NOT P
```

oder den Befehl

```
TOG P
```

benutzen. TOG steht für englisch „toggle“. Der TOG Befehl benötigt weniger Platz im EEPROM und wird schneller als die klassische NOT-P-Konstruktion ausgeführt.

Die Portvariable P darf beim TOG Befehl nur für einen einzelnen Digitalport stehen, nicht für einen Byte- oder Wordport.

- Deaktivieren eines Ports mit **DEACT**

Sobald einer Portvariablen erstmalig ein Wert zugewiesen wird, schaltet der Steuercomputer die zugehörigen Hardwarestrukturen im Prozessorchip (Transitoren) auf Ausgangsbetrieb. Es fließt also entsprechend der angeschlossenen Schaltung Strom aus bzw. in den Prozessor (max. 10 mA zulässig!). Der Befehl

```
DEACT portvar
```

deaktiviert den angegebenen Port. Das heißt, der Port wird in einen hochohmigen Zustand geschaltet und arbeitet im Eingangsbetrieb.

- der **PULSE** Befehl

Mit dem Befehl

```
PULSE portvar
```

wird ein Puls von einigen Millisekunden Breite am mit portvar bezeichneten Port ausgegeben. Steht der Port vor Ausführung des PULSE Befehls auf low (=0), wird ein High-Puls (0-1-0), ansonsten ein Low-Puls (1-0-1) ausgegeben.

Definition und Anwendung von Datentabellen

Im Standard-BASIC dienen DATA-Zeilen zum Ablegen von konstanten Datenblöcken, auf die dann sequentiell zugegriffen werden kann. CCBASIC unterstützt keine DATA-Zeilen, bietet jedoch ein weitaus flexibleres Werkzeug zur Definition und zum Zugriff auf Datenblocks. Konstante Daten können in Form von Tabellen abgelegt werden. Jede Tabelle bekommt einen Bezeichner (tablename) zugewiesen und kann beliebig viele Einträge enthalten, soweit der Programmspeicher Platz bietet. Jeder Dateneintrag (Cx) wird als Integerwert abgelegt und belegt somit zwei Bytes. Dabei können die Daten direkt im Quelltext aufgeführt werden

```
TABLE tablename C0 C1 C2 C3 ...
C4 C5 ...
... Cn
TABEND
```

oder vom CCBASIC-Compiler aus einer externen Textdatei importiert werden

```
TABLE tablename “tabfilename“
```

Die Tabellendefinitionen müssen stets am Ende eines Programms, hinter dem END Befehl stehen, da die Daten nahtlos hinter den vorangehenden Codebytes im EEPROM-Speicherchip abgelegt werden. Die Programmabarbeitung darf nie über Tabellendaten laufen, da die Daten sonst als BASIC Befehle interpretiert werden würden, was sicher zum Absturz des Systems führt.

Der Zugriff auf die Tabellendaten erfolgt mit dem Befehl

```
LOOKTAB tablename,index,variable
```

tablename bezeichnet eine gültige Tabelle, für index kann ein beliebiger Term stehen und die variable bezeichnet die Speicherzelle, in der das Ergebnis abgespeichert werden soll. Der berechnete Wert des index-Terms darf nicht negativ sein und maximal N-1 betragen, wenn die indizierte Tabelle N Einträge hat. Ergibt index den Wert 0, so wird C0 in der angegebenen Variablen gespeichert, für index gleich 1 C1 und so weiter. Folgendes Beispiel gibt den Inhalt einer Tabelle seriell aus

```
DEFINE value WORD
DEFINE i BYTE

FOR i = 0 to 3
  LOOKTAB mytab,i,value
  PRINT "mytab["; i; "]="; value
NEXT

END

TABLE mytab 12 -20 0 1000
TABEND
```

Am Bildschirm des Terminalprogramms sollte erscheinen

```
mytab[0]=12
mytab[1]=-20
mytab[2]=0
mytab[3]=1000
```

Besonders nützlich erweisen sich Tabellen beim Umsetzen von A/D-Werten in echte physikalische Größen. Eine Umsetzungstabelle hat dann in der Regel 256 Einträge. Der gemessene A/D-Wert geht dann als Tabellenindex in die Bestimmung der physikalischen Größe ein.

Zugriff auf die Echtzeituhr

Um den Stand der internen Echtzeituhr auszulesen und zu setzen, sind folgende globale Variablen definiert:

YEAR	Jahr (0...99)
MONTH	Monat (1...12)
DAY	Tag des Monats (1...31)
DOW	Wochentag (0=Sonntag...6=Samstag
HOUR	Stunde (0...23)
MINUTE	Minute (0...59)
SECOND	Sekunde (0...59)

Beachten Sie bitte, daß während des Zugriffs die interne Uhr weiterläuft. Der Sekundenwert sollte daher stets zuerst ausgelesen werden. Steht er auf 59, so muß nach dem Lesen der letzten interessierenden Zeitinformation (z.B YEAR) der Sekundewert nochmals gelesen und auf =0 getestet werden. In diesem Fall ist das Auslesen der Echtzeituhr zu

wiederholen, da eine neue Minute angebrochen ist (Extremfall Silvester mit Weiterschalten aller Stellen in Uhr und Datum).

Die Jahreszahl wird im C-Control System nur zweistellig abgespeichert. Sollte das Auswerten der Jahreszahl in Ihren Algorithmen eine Rolle spielen, beachten Sie bitte den bevorstehenden Wechsel von 1999 auf 2000, der sich im C-Control als Wechsel von 99 auf 0 darstellt.

Timer

Der interne 20-Millisekunden-Timer kann über den vordefinierten Bezeichner **TIMER** ausgelesen werden. Der Timer ist freilaufend und kann nicht gestellt oder rückgesetzt werden.

Ausgabe von Tönen mit BEEP

Die *C-Control Station* ist intern mit einem Piezo-Schallwandler ausgestattet, über den Signaltöne ausgegeben werden können. Der Befehl dazu lautet

```
BEEP ton, tTon, tPause
```

Für die drei Parameter können Konstanten oder Terme eingesetzt werden. Dabei bestimmt ton die Tonhöhe nach der Formel

ton = 250000 / freq [Hz]

tTon bestimmt die Dauer des Tons und tPause die Pause nach dem Ton. Die Einheit für die Zeitangaben beträgt 20 Millisekunden. Der Befehl

```
BEEP 568, 10, 3
```

gibt also für 10*20=200 Millisekunden einen Ton von etwa 440 Hz (Kammerton A) aus und macht danach eine Pause von 3*20=60 Millisekunden. Wenn nach einem BEEP kein weiterer BEEP folgt, kann die Pause auch auf 0 gesetzt werden. Ist für die Tonlänge 0 angegeben, wird ein Dauerton erzeugt. Der Tongenerator schaltet den Ton ein und fährt mit der Abarbeitung des BASIC-Programms fort. Mit dem Wert 0 für ton kann der Tongenerator wieder abgeschaltet werden.

Frequenzmessung mit FREQ und FREQ2

Die Frequenzmessung der *C-Control Station* basiert auf dem Pulszählprinzip bei einer Torzeit von 1 Sekunde. Die Messung erfolgt ständig im Hintergrund, parallel zur Abarbeitung des BASIC-Programms.

Die im Sekundenzyklus gezählten Impulse entsprechen also direkt der Frequenz in Hz.

Der Standard-Eingang zur Frequenzmessung (Klemme 35) kann mit FREQ2 abgefragt werden.

```
x = FREQ2
```

Er hat einen Meßbereich bis ca. 30000Hz bei einem Fehler <1%.

Ist am DCF77-Eingang keine Aktivantenne angeschlossen, so kann mit diesem Eingang alternativ eine Frequenzmessung erfolgen, deren Ergebnis mit der Funktion FREQ abgefragt werden kann.

```
x = FREQ
```

Der Messbereich reicht bis etwa 5 Kilohertz mit einem Meßfehler unter einem Prozent. Danach wird das Ergebnis zunehmend ungenauer.

- Stromsparmodus mit SLOWMODE

Anwendungen, die keine hohe Rechenleistung benötigen, können durch Aufruf des Befehls

```
SLOWMODE ON
```

den internen Takt des Mikroprozessors verlangsamen (1/16). Sollte im Verlauf des Programms wieder eine höhere Geschwindigkeit erforderlich sein, so läßt sich mit

```
SLOWMODE OFF
```

wieder der Ausgangszustand herstellen.

Achtung: Programme, die serielle Datenübertragungen verwenden, sollten den SLOWMODE nicht aktivieren, da die eingestellten Übertragungsraten mit dem Prozessortakt herabgesetzt werden.

Das Herabsetzen des Prozessortaktes führt zu einer Reduzierung der Leistungsaufnahme und kann die Akku-Überbrückungsdauer bei Netzspannungsausfall verlängern. Das macht jedoch nur Sinn, wenn gleichzeitig alle LEDs und Relais ausgeschaltet sind.

Beispiel zur Programmierung mit CCBASIC

Das folgende kleine Beispiel soll das Grundprinzip der Programmierung mit CCBASIC verdeutlichen. Die hier vermittelten Informationen versetzen

Sie in die Lage, die zusätzlichen Beispiele von der CD nachzuvollziehen und Ihren Vorstellungen anzupassen, um so schrittweise einen vollständigen Einblick in die BASIC-Programmierung der *C-Control Station* zu bekommen.

Angenommen, Sie wollen mit der Taste F1 am Modul das Relais K1 abwechselnd ein- und ausschalten. Ein Tatsendruck soll mit einem Piepton quittiert werden. Parallel dazu soll im Minutentakt das Relais K2 ein- und ausgeschaltet werden.

Starten Sie C-Control/BASIC (CCEW32D.EXE) unter Windows. Öffnen Sie eine neue Datei mit dem Menübefehl „Datei/Neu". Speichern Sie die Datei, beispielsweise unter dem Namen „TEST1.BAS".

Der Quelltext

Beginnen Sie nun, den Quelltext zu editieren.

Zunächst müssen die verwendeten Peripherieeinheiten definiert werden:

```
define K1 port[7]
define K2 port[8]
define F1 port[9]
```

Weiterhin definieren wir eine Variable, deren Verwendung noch gezeigt wird.

```
define oldsec byte
```

Ein Programm besteht meist aus Initialisierung, Hauptschleife und Unterfunktionen. In unserem Beispiel wollen wir als Initialisierungszustand das Relais K1 ein- und K2 ausschalten.

```
K1 = ON
K2 = OFF
```

Die Hauptschliefe in unserem Beispiel soll das Label „loop" tragen. Sie enthält die Abfrage der Taste F1 und die Minutentaktfunktion für Relais K2. Zu Beginn einer neuen Minute ist der Sekundenwert immer 0, für eine Sekunde lang. Um das Relais in dieser Sekunde nicht mehrmals

umzuschalten, wird die Feststellung des Minutenbeginns außerdem an den Wechsel des Sekundenwertes gebunden.

```
#loop
  if not F1 then gosub toggleK1
  if (0 = second) and (second <> oldsec) then tog K2
  oldsec = second
goto loop
```

Achten Sie auf die Negation des Tastenwertes. Da die Tasten hardwareseitig mit Pullup-Widerständen nach 5V beschaltet sind, liefert deren Abfrage im ungedrückten Zustand High- und in gedrücktem Low-Pegel! (Hinweis: die Tastenabfrage im Beispiel zur grafischen Programmierung enthält bereits die Negation).

Wenn F1 als gedrückt erkannt wird, verzweigt das Programm zur Unterfunktion toggleK1. Die Unterfunktion schaltet zunächst das Relais K1 um. Zur Entprellung und Verriegelung der Taste wird dann 100ms (5x20ms) und dann auf das Loslassen der Taste gewartet. Anschließend kehrt die Unterfunktion zur Hauptschleife zurück.

```
#toggleK1
  tog K1
  pause 5
  wait F1
return
```

Compilieren und Simulation

Wenn Sie den Quelltext eingegeben haben, compilieren Sie das Programm über den Menü-Befehl „Entwicklung/Compilieren“. Im Meldungsfenster erhalten Sie den Status der Übersetzung, gegebenenfalls eine Liste von Fehlermeldungen. Unser Beispiel sollte fehlerfrei compiliert werden, wie Abbildung 20 zeigt.

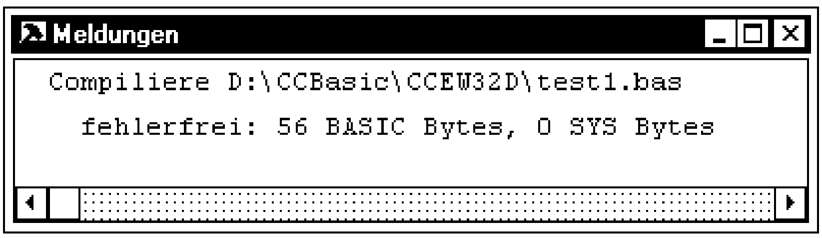


Abbildung 20

Um das Programm zu simulieren müssen wir zunächst wieder zum Quelltextfenster umschalten, entweder durch Schließen des Meldungsfensters, über das „Fenster“-Menü oder durch Drücken von „Ctrl+Tab“.

Öffnen Sie jetzt das Simulatorfenster „C-Control Digitalports“ über das Menü „Ansicht“. Im Gegensatz zur grafischen Programmierung bietet die BASIC-Programmiersoftware keine direkte Abbildung der *C-Control Station* Hardware. Das Fenster „C-Control Digitalports“ stellt alle 16 Digitalports dar. Wie bereits beschrieben, sind die Ports 1 bis 6 mit den Klemmen P1 bis P6 verbunden und über Pullup-Widerstände in der Schaltung auf High-Pegel gelegt, ebenso die Ports 9 bis 12, die mit den Folientasten F1...F4 vebunden sind. Schalten Sie im Simulatorfenster „C-Control Digitalports“ zunächst diese Ports ein, indem Sie bei gedrückter Shift-Taste auf die LEDs 1...6 und 9...12 klicken, sodaß diese hellgrün erscheinen.

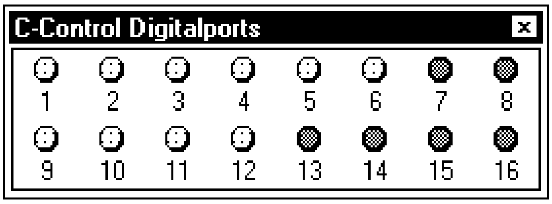


Abbildung 21

Klicken Sie dann mit der rechten Maustaste auf das Fenster, und wählen Sie im erscheinenden Menü „Als Anfangsstellung nach Reset verwenden“ (siehe Abbildung 22). Jetzt entspricht das Simulatorfenster den Hardware-Verhältnissen der *C-Control Station*. Diesen Einstellungsvorgang müssen Sie nur einmal vornehmen. Ihre Einstellungen werden beim Programmende automatisch gespeichert.

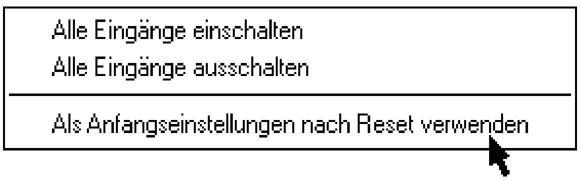


Abbildung 22

Jetzt können Sie den Simulator starten. Aktivieren Sie das Fenster Editorfenster. Wählen Sie im Menü „Simulator“ den Befehl „Ausführen“.

Die LEDs 7 und 8 erhalten einen roten Rand, die LED 7 erscheint hellrot. Das bedeutet, beide Ports werden als Ausgang verwendet, Port 7 ist eingeschaltet. Wie bereits beschrieben sind die Ports 7 und 8 mit den Relais K1 bzw. K2 verbunden. Klicken Sie nun auf die LED 9, die der Folientaste F1 entspricht. Wie beabsichtigt läßt sich der Zustand von Relais K1 umschalten.

Öffnen Sie jetzt über das Menü „Ansicht“ das Fenster „Überwachte Variablen“. Drücken Sie die „Einf“-Taste und geben Sie im folgenden Dialog „second“ ein:

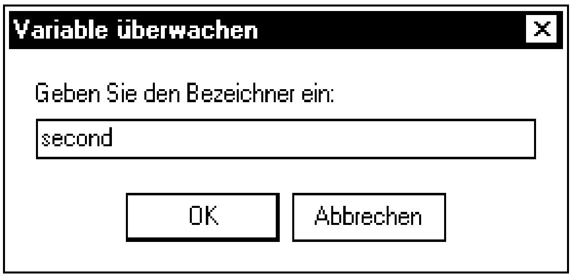


Abbildung 23

Nach dem OK sollte folgendes Fenster zu sehen sein:

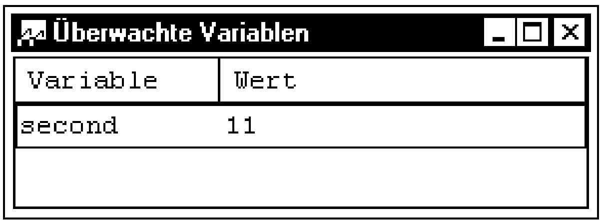


Abbildung 24

Im Fenster „Überwachte Variablen“ können Sie den Werteverlauf von System- und eigenen Variablen verfolgen, in diesem Beispiel den Sekundenwert der Echtzeituhr. Lassen Sie den Simulator eine Weile laufen. Beobachten Sie beim Wechsel der Sekunden von 59 auf 0, also zum Minutenbeginn, das Umschalten der LED 8, die den Schaltzustand vom Relais K2 darstellt.

Jetzt ist die zweite Funktion unseres Beispielprogramms nachgewiesen. Halten Sie den Simulator an, und setzen Sie ihn zurück. Die nötigen Befehle finden Sie im „Simulator“-Menü.

Übertragen und Starten des Programms

Nach dem Funktionstest kann das Beispielprogramm in die *Station* übertragen werden.

- Stellen Sie sicher, daß die *Station* in zulässiger Weise mit Betriebsspannung versorgt und über eine serielle Schnittstelle mit dem PC verbunden ist.
- Drücken Sie den Reset-Taster an der *Station*.
- Stellen Sie über „Optionen / Umgebung / Simulator und Lader“ die verwendete serielle Schnittstelle ein.
- Aktivieren Sie das Editorfenster „test1.bas“.
- Wählen Sie den Menübefehl „Entwicklung / In C-Control-Unit übertragen“.

Während der Übertragung blinkt die TX-Leuchtdiode an den RS232-Klemmen der *Station*. Im Meldungsfenster am Bildschirm werden Sie über den Erfolg der Übertragung oder eventuelle Fehler informiert. Sollte ein Fehler gemeldet werden, überprüfen Sie bitte die Verbindungsleitung, die Einstellung der seriellen Schnittstelle sowie die Spannungsversorgung der *Station*. Vor einem erneuten Übertagungsversuch müssen Sie wieder das Editorfenster aktivieren.

War die Übertragung erfolgreich, ist die *C-Control Station* jetzt programmiert.

- Drücken Sie den Start-Taster, um das kleine Testprogramm auszuführen.

Sie sehen, daß - wie programmiert und simuliert - mit jedem Drücken auf die F1-Taste am Modul das Relais K1 ein- und ausgeschaltet wird. Im Minutentakt schaltet K2 um.

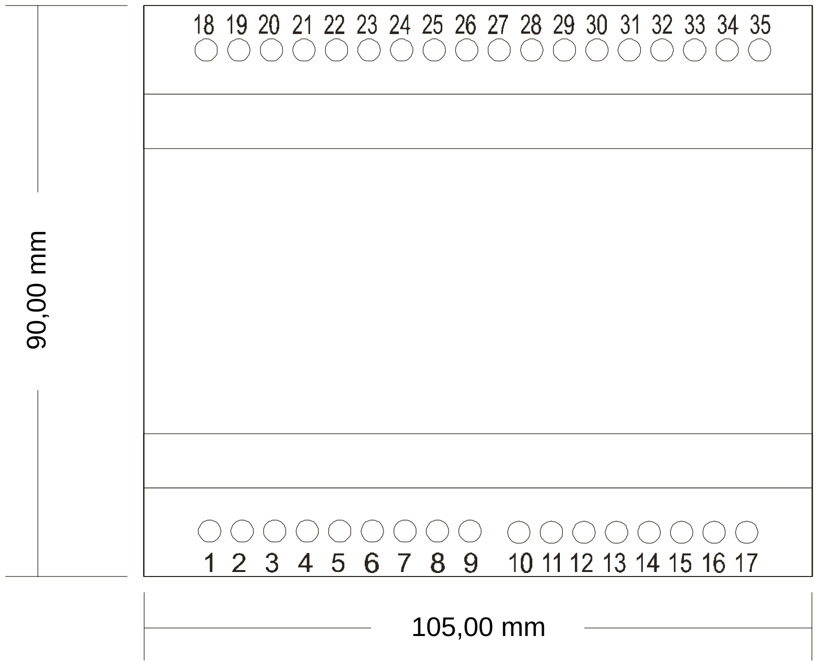
Beispiele von der CD

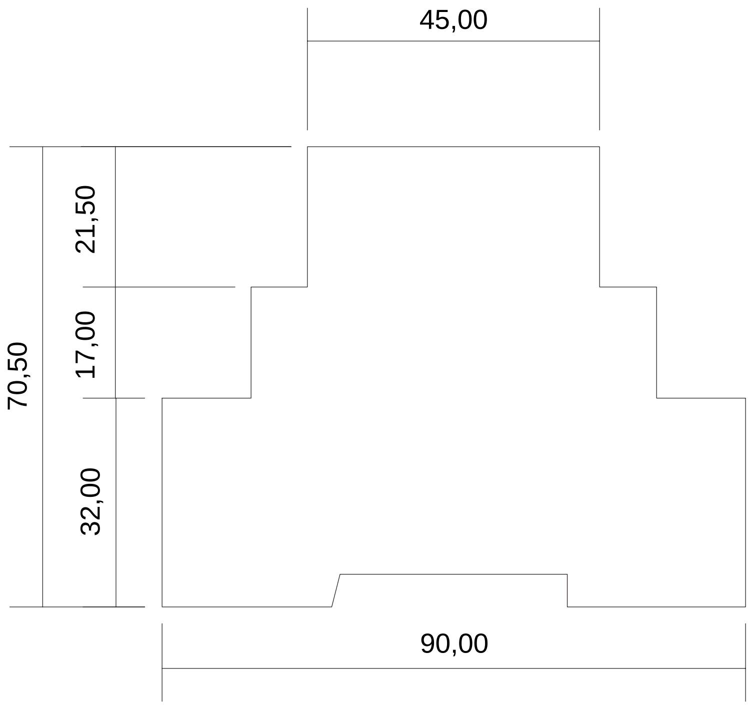
In den SAMPLES-Unterverzeichnissen der BASIC- und der grafischen Software finden Sie einige Beispiele, die Ihnen die Programmierung der *C-Control Station* deutlich machen und Lösungsansätze für zahlreiche Anwendungsfälle bieten.

Technische Daten

Betriebsspannung	230V~-Netzspannung an Klemmen 16 und 17 oder stabilisiertes 12V-Netzgerät an Klemmen 7 und 8
max. Leistungsaufnahme	2,5W
max. zulässiger Strom aus Klemme 6 (5V)	$I_{6,max}$ = 50mA
max. zulässiger Strom aus Klemme 9 (12...24V)	$I_{9,max}$ = 50mA
max. Portstrom an Klemmen 28 bis 33 (P1...P6)	$I_{28,max}$...$I_{33,max}$ = je 10mA
max. Gesamtstromabgabe	$I_{6,max} + I_{9,max} + I_{28,max} + ... + I_{33,max}$ = 100mA
Akku-Pufferstrom aus Klemme 7	ca. 5mA
Relais-Schaltleistung (K1, K2)	max. 230V~, kurzeitig 8A Einschaltstrom, max. 6A Dauerstrom
Temperaturmeßbereich (T1, T2)	-25,0°C...102,5°C 0,5K Auflösung, max. Fehler ±2,5%
Meßbereich A/D-Ports (A1...A4)	0...2,55V 0,01V Auflösung, max. Fehler ±1 Digit
Frequenzmeßbereich (Klemme 35)	1Hz...30000Hz
zulässige Umgebungs-bedingungen	0...40°C, 20...60% rel. Luftfeuchte

Außenmaße


18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
90,00 mm
105,00 mm


45,00
21,50
17,00
70,50
32,00
90,00

Impressum

Diese Bedienungsanleitung ist eine Publikation der Conrad Electronic GmbH, Klaus-Conrad-Straße 1, D-92240 Hirschau.

Diese Bedienungsanleitung entspricht dem technischen Stand bei Drucklegung. Änderungen des Gerätes in Aussehen, Technik und Ausstattung bleiben vorbehalten.

MF/CTC 29.04.98

C-Control Station

Hinweis zur Elektromagnetischen Verträglichkeit

Die C-Control Station sollte im Dauerbetrieb nicht über ein 12V-Gleichspannungsnetzgerät versorgt werden. In dieser Konfiguration ergibt sich eine erhöhte Empfindlichkeit gegenüber elektrostatischen Entladungen und Netz-Störspannungsspitzen, die zum Aussetzen des Mikrocontrollers führen können und dann einen Reset und den Neustart des Anwenderprogramms erforderlich machen.
Die C-Control Station sollte im Dauerbetrieb stets per 230V-Netzspannungsanschluß oder 12V-Akku versorgt werden.

Conrad Electronic GmbH
D-92240 Hirschau